

AD-A230 948

①
DTIC FILE COPY

SCA Reference 87-5

PERFORMANCE MODELLING OF AUTONOMOUS ELECTRO-OPTICAL SENSORS

Experimental Algorithms & Software for ATR

BIANNUAL TECHNICAL REPORT
CONTRACT NO. DAAB10-86-C-0533

June 1987

DTIC
ELECTE
S FEB 08 1991
D

Report Period
1 October 1986 - 30 April 1987

Report Prepared For
ADVANCED MODELLING TEAM
Night Vision and Electro-Optical Laboratory
Fort Belvoir, Virginia

DISSEMINATION STATEMENT A
Approved for public release
Distribution Unlimited

VIEWS, OPINIONS, AND/OR FINDINGS CONTAINED IN THIS REPORT ARE THOSE OF THE AUTHOR(S)
AND SHOULD NOT BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION, POLICY,
OR DECISION UNLESS SO DESIGNATED BY OTHER OFFICIAL DOCUMENTATION

STATISTICAL CONSULTING ASSOCIATES, INC.
P.O. BOX 2476 (148 WATERMAN STREET)
PROVIDENCE, RHODE ISLAND 02906

91 2 07 038

EXPERIMENTAL ALGORITHMS AND SOFTWARE FOR ATR

BY

ULF GRENANDER

MAY 1987



Accession For	
NTIS CRA&I	
DTIC TAB	
Unclassified	
Justification	
By <i>per Ltr.</i>	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

Table of Contents

1. Aim of report
2. Terrain generation, range data
3. Object generation, placements
4. Main algorithm for detection/recognition
5. Optimal selection of shape elements
6. Utilities
7. Experiments
8. Code
9. Manual for using code
10. Objects
11. Shape elements
12. Conclusions and continuation

Appendix 1. Micro-statistical Analysis of Noisy Images

Appendix 2. Parallel Logic Under Uncertainty, Continued and Applied to
the "Car Experiment".

1. The Task. We shall study the problem of object identification against a background of clutter and using laser radar for obtaining range data. The algorithm shall attempt to classify the object or objects in the scene. It may also be able to do the detection but this is not the primary goal since, in a multiple sensor situation, this may be better done by other sensors.

The algorithms shall be flexible so that they can easily be modified to incorporate changes in technology (as long as range data are supplied). In particular, if the accuracy of the range determinations improves drastically, only minor changes will be needed in the code.

To make this possible the code is made highly modular, so that only one, or a few, of the smaller modules need be replaced by other modules (also supplied) in addition, of course, to changing parameter values.

Our approach is actually based on the assumption that much higher accuracy in range data will soon be achieved. It should then also be possible to recognize partially hidden objects. The main ideas in the code have been chosen with this future task in mind.

Since at present we have few real images with reliable range data we have had to build a terrain-object-noise simulator to produce data on which the algorithmic ideas can be tried and which will also be useful for debugging the programs.

Simulated data are good for this purpose, but they are not enough since the model on which the simulation is based must be tested against reality. It is therefore necessary to try the programs on real and reliable range pictures when they become available, leading no doubt to modifications.

The code has been written in APL. The reason for using this unconventional programming language is that its logical structure, which is based on some of the

fundamental concepts of mathematics, makes it ideal for mathematical experimentation. The drawback is that the fact that it is interpreted, rather than compiled, makes it run much slower than, for example, optimized compiled FORTRAN code. In experiments on the computer this can be accepted - once good performance has been achieved the algorithms may be rewritten into some other language or implemented through special hardware.

2. Generation of background. The scenery will be made up of large scale features and of smaller details. The former, that can partially or wholly obscure the objects and may also sometimes confuse the algorithm by appearing 'object like', will be chosen as random elevations of fairly smooth appearance.

The centers of the elevations will form a random process in the x,y-plane on which objects will be placed. The user specifies the number of them (right argument in the functions GENTERRAIN and GENSCENE) but the locations, heights, and horizontal shape cross section are generated at random by the algorithm.

The height/width ratio can be controlled (in a probabilistic sense) by choosing a suitable power as the exponent in line [22] of GENTERRAIN. The present value is .5 which produces fairly sharp outlines, a bit like trees. Other values will produce the appearances of hills in a rolling landscape. This is also influenced by CMAX/CMIN for height distribution and by the parameters AMIN, AMAX,BMIN,BMAX that control the cross sections.

The small scale features in the background are more chaotic, at present just white noise, with standard deviation called SMALLSCALE in line [7] of GENSCENE. This could easily be replaced by a stationary stochastic process with, say, a fairly but not completely, flat spectral density.

Line [5] was inserted in order to let the user choose a particular probabilistic behavior for the sky portion of the scene. At present it is not used, setting LSKY equal to zero.

After the background is generated the program interrogates the user about placing objects (see next section), and then adds noise to the range data computed by the function SIGHT1. The latter computes the true ranges, taking possible obscurations into account and using a depression angle called PSI, assumed small.

The error level in the range sensor on the background is denoted by BACKGROUNDERROR and on the objects by OBJECTERROR. Judging by the few real pictures we have seen the former is much bigger than the latter. If this changes in the further development of laser radar some of the modules in the code should be replaced as will be discussed in section 12.

3. Generation of object shapes. The number of object shapes used, called NOBJECTS, was only six in the following experiments but can be set by the user to any value he wishes.

To prepare for bigger values and to facilitate entering the shape we have prepared the functions GENOBJECT, COLLECT, INSIDE, SCALING. They enable the user to present any object of polygonal shape: he is prompted by the program to type the coordinates of the vertices and the programs do the rest. For more details in using these and other programs, see section 8.

In the experiment we used the object shapes shown in section 10, representing front and side view of a tank, front and side view of a small stylized house, and two side views of vehicles. We used polygons with few vertices, which may have been psychologically wrong, since the appearance of the object boundaries will be too straight and simple. In future experiments one should use

more detailed polygonal or spline representations of the shapes - this will also make it easier for recognition algorithms.

The program PLACE places objects in the x,y plane at desired location for the midpoint of the lower, horizontal part of their boundary and at height 2=0. This implies that height variation in the background (vegetation, small hills, etc.) will normally hide the lower horizontal part of the boundary which therefore will not help in discrimination. This effect was not planned but showed up in many of the experiments. It is not clear if this is a realistic effect.

4. Main approach. The logical organization is based on ideas presented in the two working papers enclosed as Appendix 1 and 2, but modified in two respects:

- (i) we now assume range information rather than IR or visible light images,
- (ii) we assume low accuracy (at least for the time being) that seems to rule out the sophisticated algorithms that was proposed in Appendix 2.

We are preparing for the time when improvements in the accuracy of range data will be made, enabling us to deal with obscuration. This of course implies that algorithms must be based on local features, since global ones may be hidden from sight.

We shall think of a scene as presented in a deformed image

$$I^D = D(R[\sigma(g_0, g_1, \dots, g_{n-1})])$$

Here the generators g_i are local features, actually arc elements making up the parts that can be seen of the boundaries of the objects in the scene. The connector graph σ joins arc elements in the appropriate order along the (partial) boundaries of each object. Thus σ will not be connected if several objects are present in the scene.

The operator R takes the configuration $\sigma(g_0, g_1, \dots, g_{n-1})$ into a set: the part of the scene consisting of objects as it appears in the image plane of the sensor.

The deformation mechanism D takes the set mentioned and transforms it into range data adding observational noise.

Our task is to recognize the objects and this will be done in terms of the generators g_i and the connector graph σ . In other words we need estimators g_k^* of arcs g : generator estimates. We shall do this by choosing a collection $\{g_k^*\}$ of shape elements, each consisting of a small binary picture, a template, say of size $LTEMPX$ by $LTEMPY$. In the experiments below we chose $LTEMPX = LTEMPY = 11$ but this may not have been a wise choice.

We postpone the important problem of how to select $\{g_k^*\}$ until section 5. If we tried to superimpose each g_k^* at each pixel in our image, and then test for goodness of fit, the results of the tests should tell us a good deal about hypothetical locations of objects.

But this is clearly wasteful, since only a small part of the picture will normally belong to object boundaries. Instead we proceed in steps as follows to reduce CPU time.

4.1. Since $BACKGROUNDERROR \gg OBJECTERROR$ the local variation in grey level is much smaller on the objects than off; this is seen clearly in the real pictures.

Exploit this by finding, at each pixel, if the variance of the pixel values at the $1+8$ values (8 neighbors) is large or small. Keep only the latter ones. This is very fast and is done by the function CHANGE as called in statement [10] of TESTS1.

4.2. We now have a binary image. To remove from consideration isolated or nearly isolated points in the set defined by the binary image = 1, we apply CLEAN which eliminates points with less than FEW neighbors. This is also very fast.

4.3. Compute the boundary of the binary image, applying the function IBOUNDARY. The resulting, much thinner, binary image is then represented as a 2-column matrix with x and y coordinates of the boundary obtained; this is carried out by the function COORD in line [11] of TESTS1. This is even faster.

4.4. We now move the shape elements g_k^* to each point remaining in the binary image after Step 4.2 and test the fit by computing the (counting) measure of the symmetric difference

$$m_k(x,y) = m(g_k^*, \text{subpicture at } x,y).$$

This is done in statements [14]-[27] of TEST1. Here NSLICE denotes the range of k in $\{g_k^*\}$ and we sweep the boundary completely, except when we are very close to the border of the image where room is not available for placing any shape element g_k^* : statements [12]-[18] take care of that exception.

In [11] we compute a crucial parameter, THRESHOLD which is proportional to the size of any shape element, $LTEMPX \times LTEMPY$. The proportionality constant, called K, is crucial. We have used values in the range 80% - 95%. It determines whether

$$m_k(x,y) > K \times LTEMPX \times LTEMPY$$

when we accept the shape element temporarily for further hypothesis testing, or not.

This step is time consuming and takes a good deal of the whole CPU time used by the driver RECOGNIZE.

The result of TESTS1 is a vector TESTSEL, of k-values for temporarily accepted shape elements, and a 2-column matrix TESTSLOC of their locations.

4.5. We now try to reconcile the information in the test results TESTSEL and TESTSLCC with the corresponding lists associated with each hypothetical object. In Appendix 2 we suggested that this matching be done by stochastic relaxation, a suggestion that has not been implemented. We believe that in the future, when advances in sensor technology will support more refined methods, that may be the way to go. At present we are rescued by the fact that we need not do subgraph matching, which is notoriously hard to compute. Instead we match subsets of two geometries.

Each of the points (x_j, y_j) in TESTSLOC carries a label k_j of the shape element $g_{k_j}^*$ that it supports. Similarly at each location (u_i, v_i) of the hypothesis lists HYPLOC there is a label ℓ_i of its shape element $g_{\ell_i}^*$. We want to match the relative positions

$$(x_j, y_j) = (u_i, v_i) + \text{offset}.$$

We shall do this by brute force, trying all combinations for the offset. This is clearly wasteful and we shall return to this later.

For each hypothesis the function FULLTEST tries all combinations for offset, lines [8]-[9] keeping track of the number of agreements obtained and saving the largest number. We also compute a vector KEEP intended for later use to successfully remove elements from the testlists; this has not been implemented yet however.

This step is the most time consuming in the whole algorithm, especially if K is small like 80% when the testlists tend to become excessively large.

4.6. The rest of RECOGNIZE collects the accumulated evidence and prints, either nothing when no object has been detected, or, in the opposite case a list of possible objects and their hypothetical locations, lines [15]-[26], in decreasing order of credibility. The number of agreements found between subset of the testlist and a subset of a hypothesis list is printed as the credibility. Very unlikely hypothetical objects are not printed.

In the present form of the algorithm several objects are allowed in the scene, but not more than one occurrence of each object type. This restriction can be removed by using the vector KEEP mentioned above.

In lines [27]-[40] hypotheses are removed that would involve centering objects so close that it is physically impossible. The minimum distance is a global variable called MINDIST.

5. Optimal Selection of Shape Elements. In the early stages of developing the program TESTS1 and its forerunner TESTS we used arbitrarily selected shape elements where the sets were chosen as rectangles, triangles, and squares with different locations and orientation in the LTEMPX \times LTEMPY square.

This seemed to work fairly well, but TEST1 often missed to fit the shape elements even in cases where the picture, displayed on the monitor, looked clear enough. This is the reason we tried to choose the shape elements in a more systematic manner.

Given the shape element, to use it as an estimator of location is not a new mathematical topic. It was studied in U. Grenander (1978): Pattern Analysis,

Springer-Verlag, section 5.1, although with a different deformation mechanisms. That sort of theoretical analysis ought to be carried out for the present situation, since it may lead to better microstatistics, see Appendix 1.

Let X be the space of all possible shape elements, and say that we are given a very large sample x_1, x_2, \dots, x_n from some probability distribution P over X . Of course X is a discrete space, but of enormous cardinality, in our case 2^{121} , so that we shall allow ourselves to treat P as given by a ('coarse grained') density f , in some vague sense continuous. We would like to select $y = (x_{v_1}, x_{v_2}, \dots, x_{v_p})$ from (x_1, x_2, \dots, x_n) where $p \ll n$. The x_{v_i} will serve as our shape elements, and in order that they serve well, we should let x_1, x_2, \dots, x_n be all or most of the shape elements that occur for the given objects.

But that will lead to excessive CPU time requirements: we cannot test all the shape elements x_i . Instead we use only the selected ones and hope to cover the most important part of X , namely the Lebesgue set

$$L(c) = \{x \mid f(x) \geq c\} \subset X.$$

To estimate the set $L(c)$ is a problem in abstract inference and closely related to Theorem 4.1 in section 5.4 of Grenander (ibid.). We shall use the following procedure.

Choose a radius r , (denoted by the global variable RADIUS in the code) and solve

$$\max_i N(x_i, r); \quad N(x_i, r) = \#\{x_j \text{ in ball centered at } x_i \text{ and of radius } r\}$$

Select x_{v_1} where v_1 realizes the max. Remove all x 's from that ball, maximize the N -function, which is proportional to a coarse grained density, giving x_{v_2} and continue until p shape elements have been selected.

Remark. It seems likely (but not proven) that this can be formalized by a theorem saying when and how ~~the~~ estimates the Lebesgue set in a general situation. It is actually a sieve estimator and can be expected to be consistent if we let $p \rightarrow \infty$ slowly compared to $n \rightarrow \infty$ and simultaneously $r \downarrow 0$.

This is implemented in two steps.

5.1. We first form the long list $\{x_i\}$ by calling the program FINDGENS. It operates on the 3-dimensional array ALLOBJECTS containing all the object shapes, and takes all the shape elements separated by a distance called DIST along the respective boundaries.

It calls the auxiliary functions SELECT and SCREEN, that separates points along the boundary, and the function NINBALL which computes the coarse grained density.

5.2. The second step is to do the estimation of the Lebesgue set, which is implemented by the function LEBESGUE that takes as input the global variable SLICES created by

SLICES ← FINDGENS

It does what was described above, but in addition it deletes shape elements that contain too many zeroes or too many ones. The reason for this is that such extreme shape elements have too low inferential power as estimators; it is done in statements [18]-[19].

This construction of an optimal set of shape elements requires a lot of CPU time, but can be done off line.

6. Utilities. When experimenting with the recognition algorithms it is useful to display pictures, say of small $LX \times LY$ dimension, quickly even if crudely. This is done by the function display.

Since matrix coordinates are oriented differently from the usual Cartesian system we should apply TURN to a matrix before displaying it.

To see a binary matrix directly do

SEE MATRIX

To scale a digital picture from level 0 to NLEVEL do

IMAGE ← NLEVEL SCALE IMAGE.

Some statistical tests have been implemented for use in the early version TEST and RECOGNIZER

STUDENT:	Student's two sample test
FISHER:	Fisher's Z-test
KOLMOGOROV:	Kolmogorov-Smirnov's two sample test

To display all the selected shape elements execute SEEGENS.

The function WHITE produces a field of standardized white Gaussian noise. Since memory limitations prevented us from doing this by a fast APL algorithm based on 3-dimensional arrays it has been done by looping and is therefore quite slow.

Remark. This approach shares some features with the other recognition algorithm that our group at Mathematical Technologies Inc. (Brown University and University of Massachusetts) has been developing. In particular they both use some sort of local test that are ultimately combined to make the global decision about objects in the picture.

The present algorithm differs from the other one in that it is based on computed shape elements, it avoids a complete scan by shape elements over all or most of the picture, and the details of the final matching - the decision function.

In TESTS we employed brute force computation for finding matches between the pre-computed shape elements and the observed picture. It called the function VIEW (or alternatively VIEW1 for a correlation test) which sweeps the entire picture with a step called STEP. It was found necessary to make STEP small, at most = 4, better = 2, to get acceptable performance. This step in the preliminary version of the decision function required massive CPU time.

It was followed by executing SEARCH, which finds all the major local maxima, separated by a distance called DIST. This step is fast.

To get better performance TESTS was replaced by TESTS1, to be described below. This not only improved the performance but reduced CPU time by a factor 10.

7. Experiments. We now begin experimenting with the code for detection/recognition. The first, very preliminary experiments, used RECOGNIZER and TESTS employing one of three statistical tests.

It was not very surprising that STUDENT did not give good performance. After all the model used in the simulator tries to mimic one aspect of the current state of laser radar technology: the signal from the background is very noisy. Hence a test for distinguishing the two mean levels in the on/off parts of the shape element can be expected to have low power.

On the other hand we had hoped that the Kolmogorov-Smirnov test would perform well. This was not the case, however, for reasons that are not yet well understood.

Fisher's Z-test, for distinguishing between the two variances in the on/off parts was much better. It required a lot of CPU time, however, at least in the interpreted APL code, so that we therefore modified the algorithm to the one in RECOGNIZE and TESTS1. We shall not report the preliminary experiments: all the following ones use RECOGNIZE.

The experiments were coded as follows.

EXPi means no object present, POWER=i

EXPji means object no. j present, POWER=i

EXPDKhji means objects k and j present, POWER=i .

The letter D stands for double. If the letter M occurs it signifies that CMAX=20 rather than the default value 10, leading to larger obstructing features in the landscape. If the code ends with A,B, or C it means an earlier experiment has been replicated but with NHILLS larger than the default value 4.

7.1. We begin by testing the detector on scenes without any objects.

EXP3.

$A, B, C =$
5.83505516 16.77435588 3.342535154
 $X0, Y0 =$
121 86

$A, B, C =$
4.9306128 16.81285364 1.458086326
 $X0, Y0 =$
79 39

$A, B, C =$
12.7033143 25.02723766 2.069762536
 $X0, Y0 =$
84 26

$A, B, C =$
5.54067033 27.10408122 1.029860429
 $X0, Y0 =$
117 121

The features are quite small and are barely visible in the picture:



The algorithm prints nothing; no detection is claimed.

EXP3MA. Here NHILLS=6 with the result

A.B.C =
8.87239012 20.2616818 16.1133888?
X0,Y0 =
69 43

A.B.C =
8.39912928 22.17009896 4.543721174
X0,Y0 =
86 71

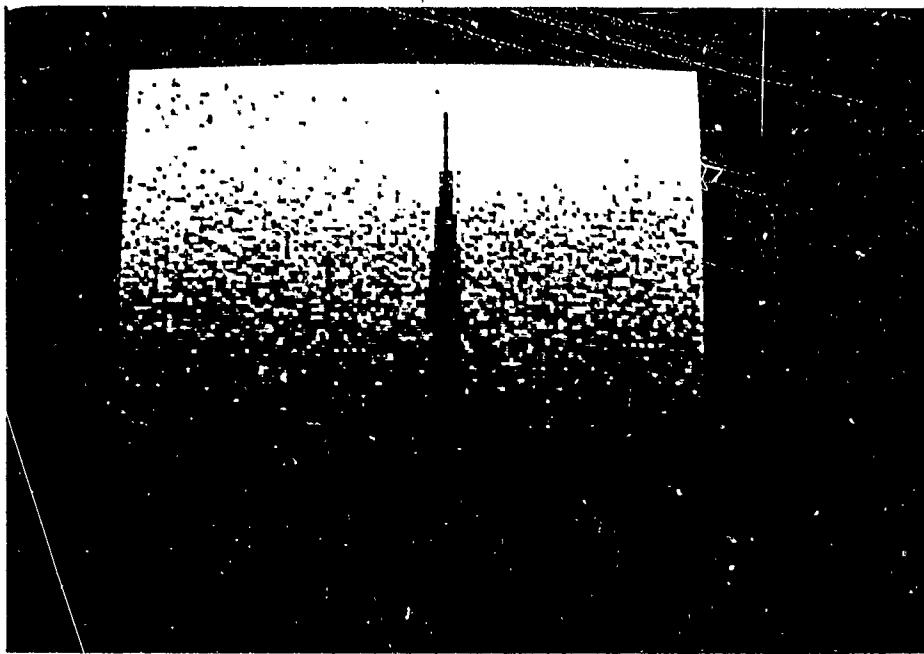
A.B.C =
9.51475366 26.46615592 6.021179599
X0,Y0 =
66 2

A.B.C =
13.9443201 10.1917297 3.063506822
X0,Y0 =
98 63

A.B.C =
10.49514866 19.96723674 4.910012924
X0,Y0 =
95 28

A.B.C =
28.01719476 10.99406942 1.039744843
X0,Y0 =
60 96

and one can see the biggest one (the first above) clearly in the picture:



No detection claimed by RECOGNIZER.

EXP3MB. With NHILLS=8, otherwise the same, we get

A.B.C =
23.82277362 27.35850764 3.081175733
X0,Y0 =
75 80

A.B.C =
4.9748388 8.11624798 12.16930329
X0,Y0 =
38 93

A.B.C =
24.90576228 29.1501744 6.2963415
X0,Y0 =
5 61

A.B.C =
21.61912854 14.69708094 12.26605857
X0,Y0 =
107 109

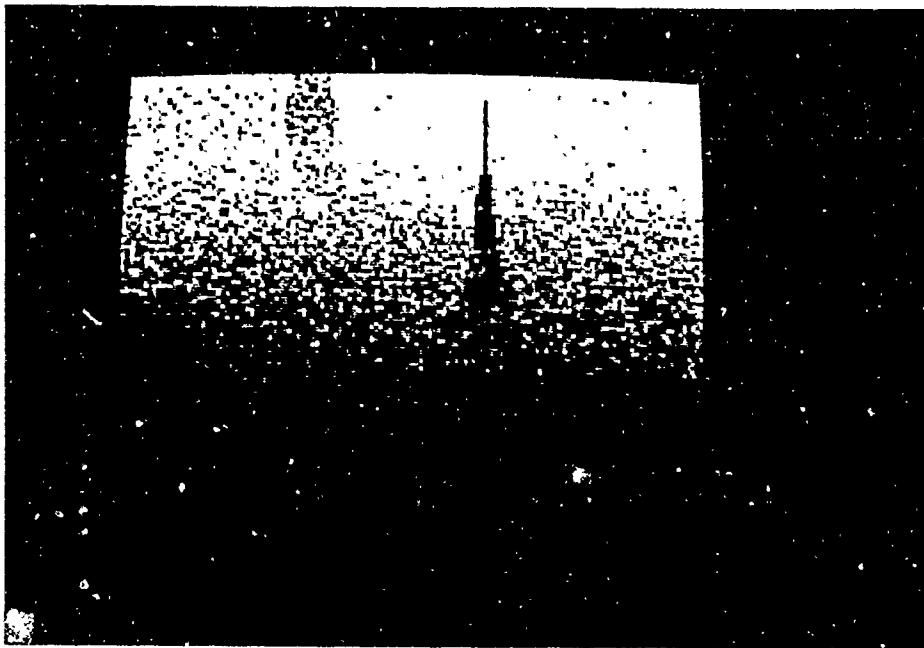
A.B.C =
25.94169692 20.10258286 1.010208911
X0,Y0 =
46 71

A.B.C =
8.21223868 27.09738418 7.034014214
X0,Y0 =
98 81

A.B.C =
17.71027268 20.5557171 19.87010288
X0,Y0 =
78 53

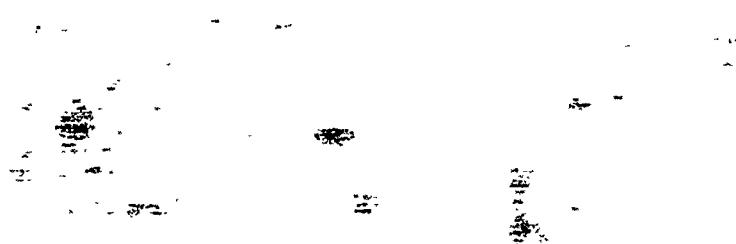
A.B.C =
29.47582726 8.23294846 1.398242615
X0,Y0 =
126 86

and the two biggest features are seen in the simulated background



No detection claimed.

EXP1MA. We now lower the exponent in the power law for the probability distribution for the height of the landscape features. This will produce a more irregular terrain, as is seen in



A.B.C =
6.68336848 19.37728582 5.41831339
X0,Y0 =
44 31

A.B.C =
27.18718064 22.94571718 19.03038763
X0,Y0 =
38 46

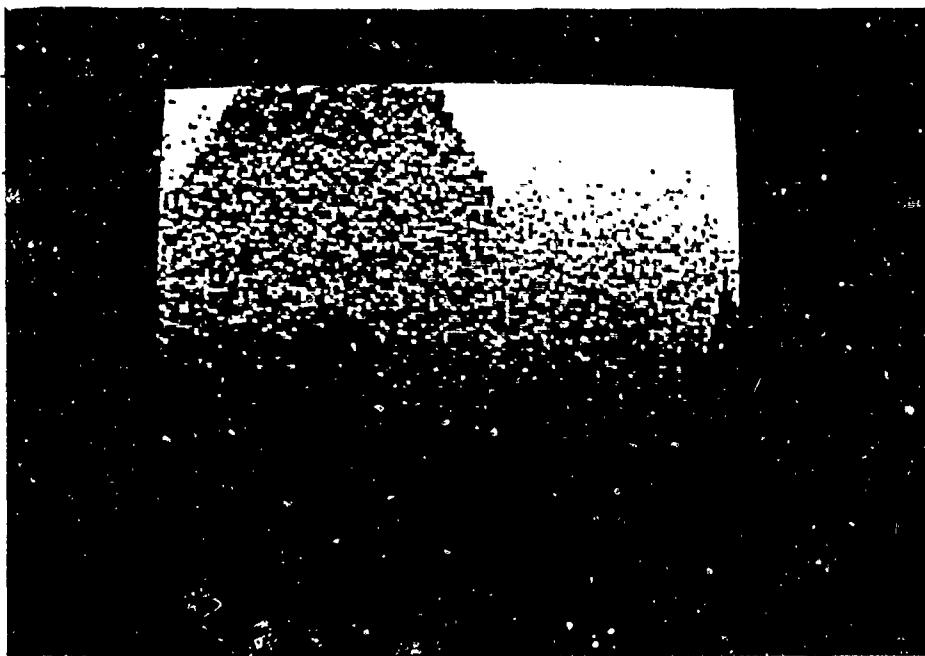
A.B.C =
13.46441574 4.43925466 18.94557258
X0,Y0 =
35 74

A.B.C =
5.7185285 27.30880734 7.66947937
X0,Y0 =
87 114

A.B.C =
24.27206324 12.56854284 18.17311181
X0,Y0 =
124 12

A.B.C =
5.96815684 10.81602766 1.71429037
X0,Y0 =
108 122

We see three big features, one dominating the scene:



No detection is claimed, although one, fairly small, feature resembles object no. 1.

Finally,

EXP1MB. With NHILLS=8 gives an even more irregular landscape

A.B.C =
10.90104948 29.9422618 13.8577919
X0,Y0 =
93 122

A.B.C =
15.00888204 14.28380626 14.10716026
X0,Y0 =
40 28

A.B.C =
18.023165 27.33604338 19.1855004
X0,Y0 =
65 71

A.B.C =
19.9293719 6.9545074 17.37612698
X0,Y0 =
125 2

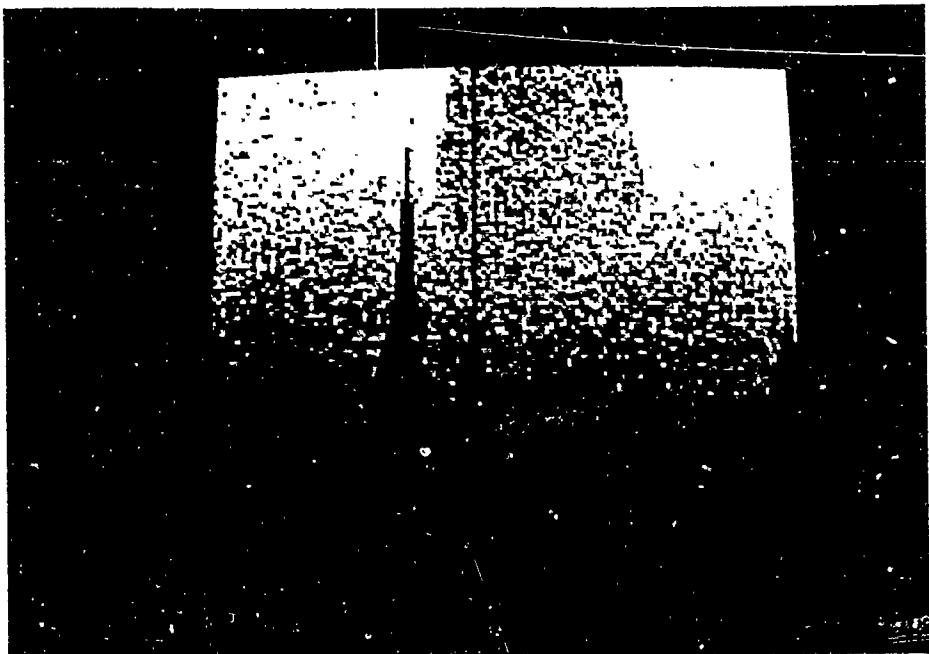
A.B.C =
7.08649952 8.79826464 14.47271931
X0,Y0 =
87 80

A.B.C =
11.9200693 22.60778608 10.5474088
X0,Y0 =
55 46

A.B.C =
10.73716446 5.52556014 3.98883433
X0,Y0 =
110 64

A.B.C =
26.45206262 17.81994588 11.10730498
X0,Y0 =
90 25

in which we can see several features, one broad and high, another very thin



No detection made.

In these, and many other experiments with no object, we found no case with false alarm. This is not necessarily good since it may mean the algorithm is too conservative: it may miss some objects when actually present. We should keep this in mind later on.

7.2. We now place an object in the scenery and will look at what happens for the six object shapes we have used (see section 10 for their appearance).

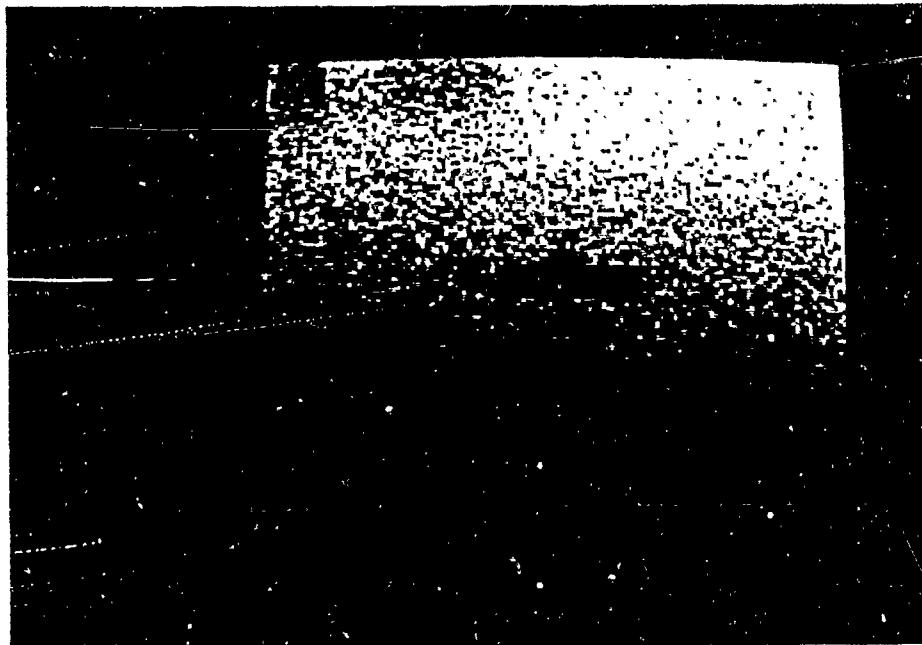
EXP01M. With NHILLS=4 here and in most of the following experiments we get, after placing object no. 0 in the middle of picture:

$\lambda_{\text{obs},C} =$
.578044.0 11.10851580 18.81710216
 $\lambda_{\text{obs},H} =$
11.14

$\lambda_{\text{obs},K} =$
11.80171000 21.38100116 7.17170050
 $\lambda_{\text{obs},L} =$
07 10

$\lambda_{\text{obs},M} =$
23.400630.5 21.3331437 17.46970080
 $\lambda_{\text{obs},N} =$
8 50

$\lambda_{\text{obs},O} =$
23.7135007 17.70724446 13.60881120
 $\lambda_{\text{obs},P} =$
125 9



With an earlier form of RECOGNIZE we get the decision

RECOGNIZE EXP01M
ALGORITHM DECIDES CREDIBILITY OF HYPOTHESES:
OBJECT NO. 0
AT LOCATION 68 70
TO DE = 33

which is the correct one.

EXP01. This is very similar, but CMAX has been lowered to 10 so that landscape features are less visible:

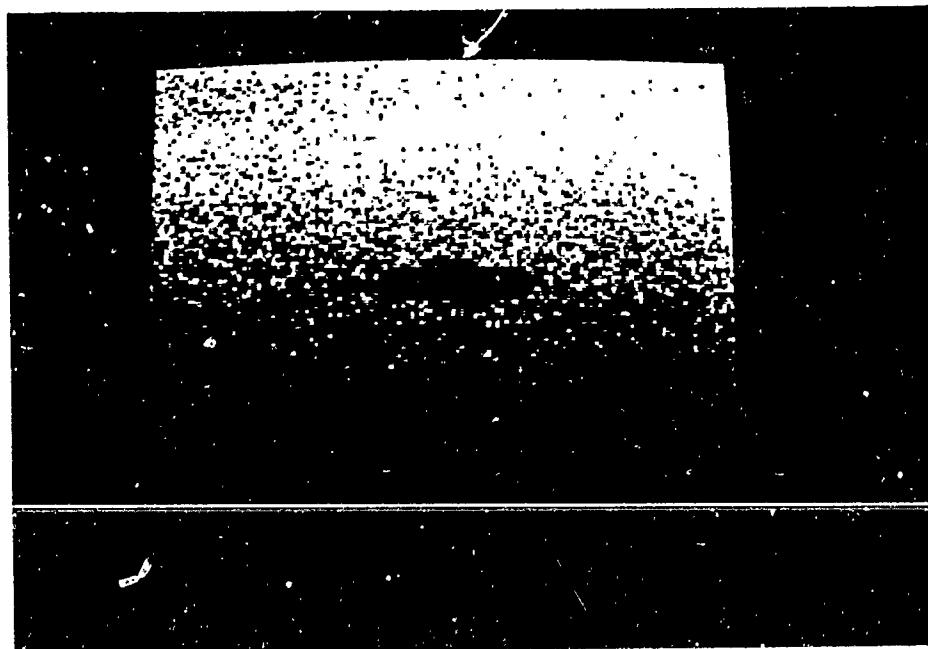
$A, B, C =$
23.9604834 9.015423 1.73459750
 $X0, Y0 =$
24 69

$A, B, C =$
22.31231974 17.16096704 5.97615203
 $X0, Y0 =$
38 42

$A, B, C =$
10.37310900 22.36147973 5.43232636
 $X0, Y0 =$
13 31

$A, B, C =$
13.26480204 29.53145816 2.11446029
 $X0, Y0 =$
26 16

with the simulated range picture looking like:



The algorithm makes the right decision:

RECOGNISE EXP01
ALGORITHM DECIDES CREDIBILITY OF HYPOTHESES;
OBJECT NO. 0
AT LOCATION 57 70
EG EE = 24

We now place object no. 1 in the landscape. It is smaller and hence more difficult to detect/recognize.

EXP11. With POWER=11, NHILLS=4 the simulator produces main features

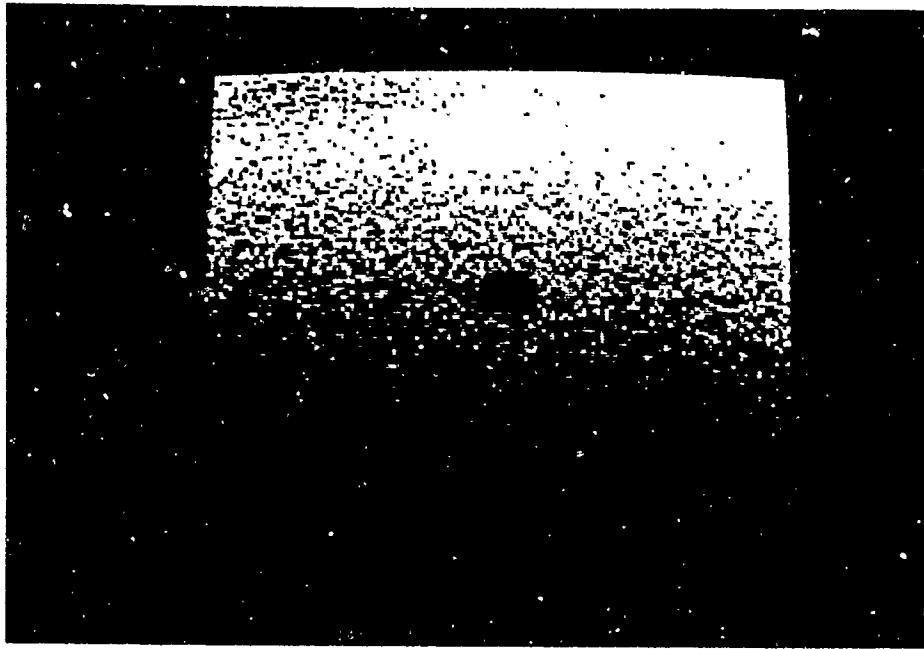
A,B,C =
26.12466938 28.79870164 6.46271659
X0,Y0 =
40 92

A,B,C =
10.22667578 5.7426149 5.19898006
X0,Y0 =
46 14

A,B,C =
29.39374266 6.63699956 6.55634035
X0,Y0 =
20 112

A,B,C =
25.40977358 24.06750928 1.91069210
X0,Y0 =
85 28

resulting in a picture like



which is correctly identified by the algorithm:

RECOGNIZE EXP11
ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 1
AT LOCATION 61 69
TO BE = 18

This is satisfying but it should be observed that this was done with CMAX=10 (no M at end of name of experiment) so that there was no big features to confuse the decision function.

EXP22. Placing object no. 2 and with power level 2 in the exponent the simulator gives

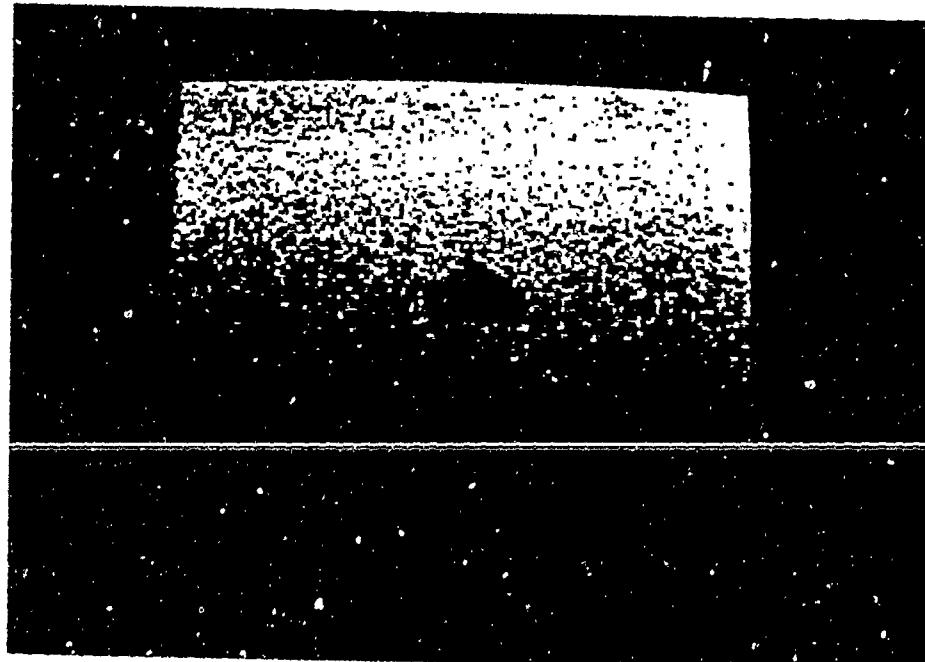
EXP22<GEN:SCIPY 4
A,B,C =
27.72161632 9.21210244 1.713029502
X0,Y0 =
41 121

A,B,C =
6.0031674 27.23793464 3.910577603
X0,Y0 =
74 65

A,B,C =
10.30703116 8.9036494 7.233083104
X0,Y0 =
57 110

A,B,C =
19.0696473 17.16288792 6.60136142
X0,Y0 =
64 43

looking like



and with the decision

RECOGNIZE EXP22
ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 2
AT LOCATION 64 69
TO BE = 17

Now we make it harder for the decision function by choosing CMAX=20,
POWER=1. Placing object no. 3

EXP31M. The simulator generators

EXP31M<GENSCENE 4
A,B,C =
7.02304912 8.39026986 7.83416738
X0,Y0 =
46 123

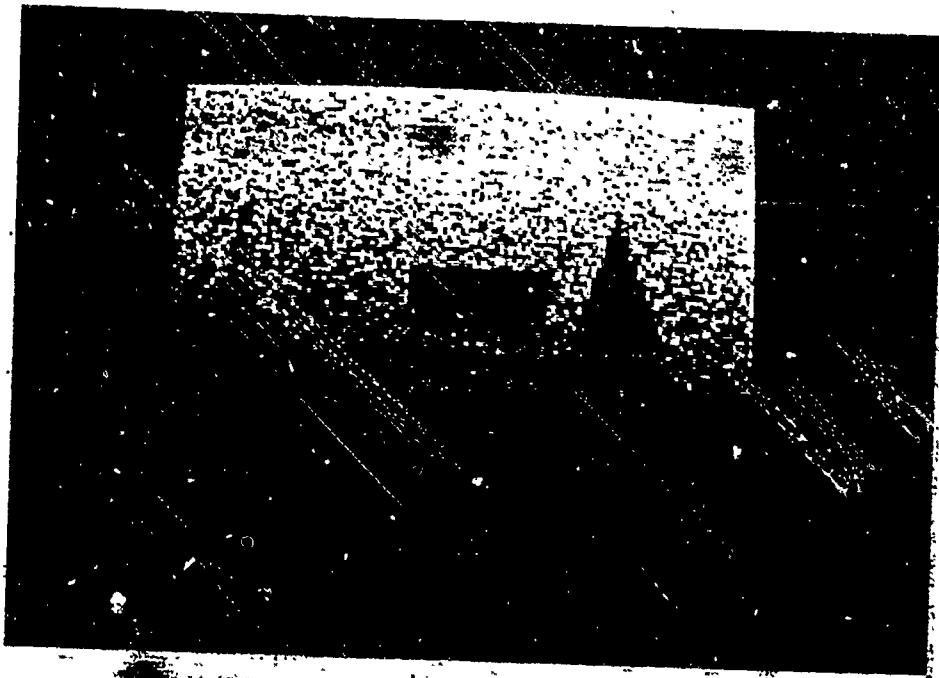
A,B,C =
29.42741812 26.61937834 14.07600349
X0,Y0 =
96 41

A,B,C =
12.57944932 28.80842312 15.01017801
X0,Y0 =
13 48

A,B,C =
13.51163376 21.3920344 12.63808995
X0,Y0 =
103 4

PLACE OBJECTS?
ANSWER 0 OR 1
□:
1
LOCATION ?
□:
64 64

OBJECT ?
□:
ALLOBJECTS[3;;]
PLACE OBJECTS?
ANSWER 0 OR 1
□:
0



and the decision function gives

RECOGNIZE EXP31M
ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 3
AT LOCATION 65 77
TO BE = 43

EXP41M. With object no. 4 the simulator produces a very 'hilly' landscape

RECOGNIZE EXP41II
A,B,C =
10.74735123 20.73325782 15.51231480
H,G,I,O =
30 38

J,K,L,M,C =
27.34530204 12.24026242 14.22266672
Y,G,Z,C =
35 27

N,O,P,C =
8.26303486 22.82975362 19.94305327
H,G,I,O =
21 37

PLACE OBJECTS?
ANSWER 0 OR 1
?:

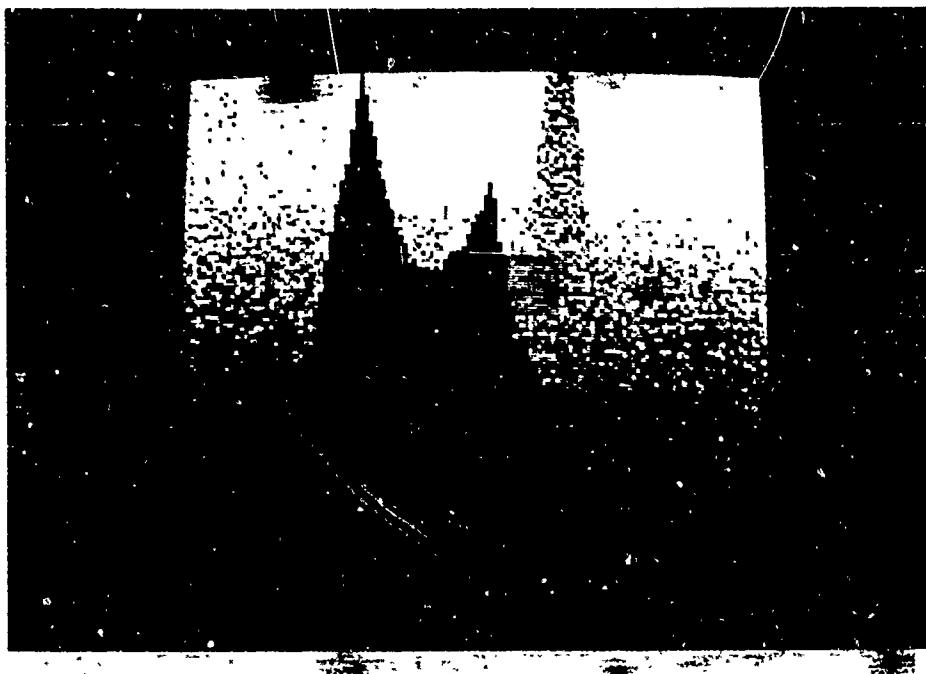
LOCATION ?
C:
64 64
OBJECT ?
C:
ALLOBJECTS[4;;]
PLACE OBJECTS?
ANSWER 0 OR 1
C:
0

with the decision

RECOGNIZE EXP41II
ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 0
AT LOCATION 70 71
TO BE = 6

This is wrong, the object is no. 4. The reason for the mistake is obviously that too much of the object has been hidden by one of the main features in the landscape. With the current noise assumptions the recognition algorithm is not able to cope with this amount of obscuration.

Detection was achieved correctly.



EXP43. To see how the recognition algorithm handles the shape of object no. 4 in a less chaotic background we lower CMAX to 10, momentarily, and get

-31-

EXP43<OPTIMIZE "

A,B,C =
25.3950016 24.92510316 1.002019302
X0,Y0 =
51 74

A,B,C =
0.40005172 20.3029607 3.043787000
X0,Y0 =
111 13

A,B,C =
23.074445 0.20003324 0.960723366
X0,Y0 =
66 96

A,B,C =
9.96226514 7.79372005 1.330301462
X0,Y0 =
126 43

PLACE OBJECTS?

ANSWER 0 OR 1

1
LOCATION ?

64 64
OBJECT ?

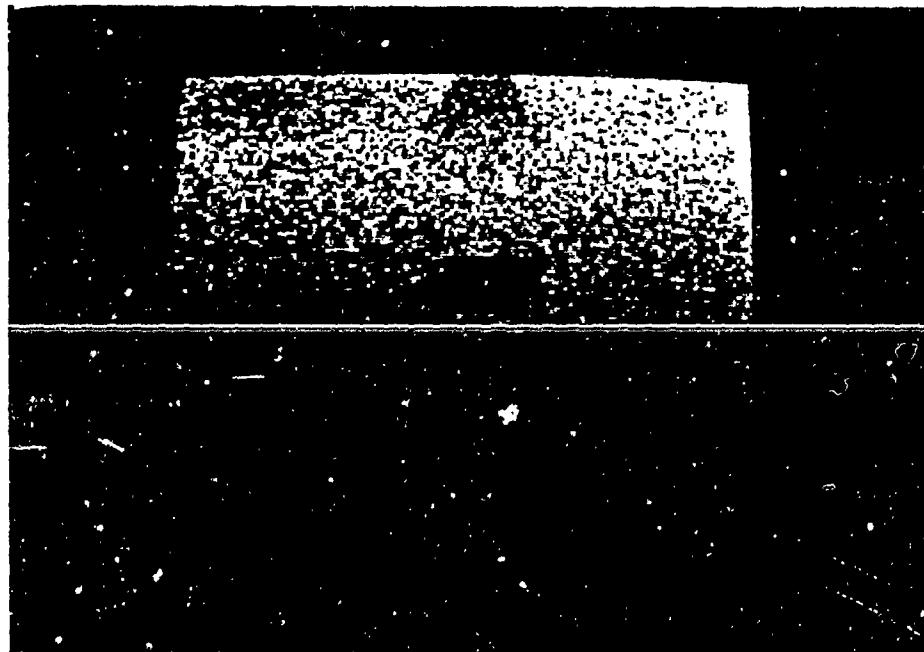
ALLOBJECTS[4;;]

PLACE OBJECTS?

ANSWER 0 OR 1

0

looking like



and the decision function then succeeds:

RECOGNISE EXP43
ALGORITHM DECIDES CREDIBILITY OF HYPOTHESES:
OUTLET NO. 4
AT LOCATION 68 74
SO PE = 37

EXP51M Finally object no. 5.

EXAMIN-GENSCENE
 $A, B, C =$
25.55025644 4.552795504 17.14262638
 $X0, Y0 =$
62 115

$A, B, C =$
19.83151518 26.2770522 8.61500202
 $X0, Y0 =$
9 110

$A, B, C =$
19.68181472 6.26387694 9.0260035
 $X0, Y0 =$
31 57

$A, B, C =$
19.75220166 19.25732936 14.10708654
 $X0, Y0 =$
32 52

PLACE OBJECTS?
ANSWER 0 OR 1

:
1
LOCATION ?

:
64 64
OBJECT ?

:
ALLOBJECTS[5;;]
PLACE OBJECTS?
ANSWER 0 OR 1

:
0

with the appearance



Note that the object is again obscured partially

RECOGNIZE EXP51M
ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:

OBJECT NO. 1

AT LOCATION 56 69

TO BE = 14

ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:

OBJECT NO. 2

AT LOCATION 56 69

TO BE = 14

ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:

OBJECT NO. 5

AT LOCATION 55 69

TO BE = 14

This means that the algorithm could not make up its mind which of objects 1,2, and 5 to choose.

We have found that objects 2 (small) and 5 (similar) are sometimes not successfully identified when obscured.

7.3. Now we shall place two objects in the picture. The performance of the algorithm will no longer be as good as before.

EXPD042: The simulator gives

EXPD042<GEN&CV>
A,B,C =
3.713358 25.40052064 0.826041633
X0,Y0 =
59 68

Z,ROT =
23.9322126 21.7616906 2.76767361
X0,Y0 =
88 34

A,B,C =
15.03075558 17.91106704 2.701454586
X0,Y0 =
56 62

A,B,C =
16.10860204 17.20951410 1.114082528
X0,Y0 =
57 52

PLACE OBJECT?
ANSWER 0 OR 1
0:

1
LOCATION ?
0:

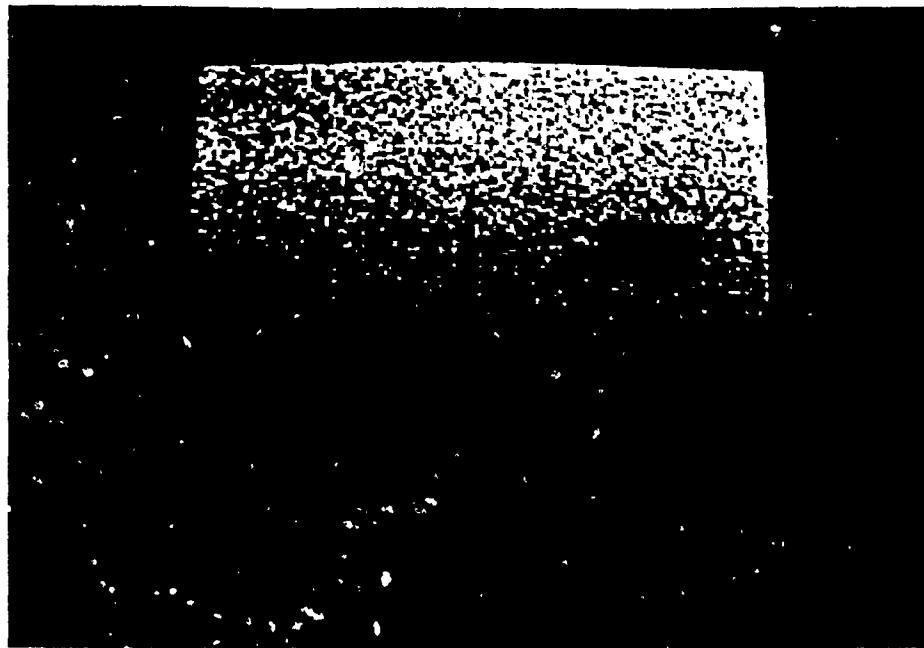
37 37
OBJECT ?
0:

ALLOBJECTS[0;;]
PLACE OBJECTS?
ANSWER 0 OR 1
0:

1
LOCATION ?
0:

95 70
OBJECT ?
0:
ALLOBJECTS[4;;]
PLACE OBJECTS?
ANSWER 0 OR 1
0:

looking like



But

RECOGNIZE EXPD042
ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 4
AT LOCATION 96 83
TO BE = 27

so that object no. 4 is correctly identified, but object no. 0 is not even detected. The reason for this is unclear. The lower horizontal boundary of it is hidden by low level obscuration, but this is always the case in these experiments, and does not seem to seriously affect the power of the detection/recognition function.

EXP133. Here the simulation gives

A,B,C = EXP133-OBJSIM7 4

X0,Y0 = 7.30377113 26.51979912 1.289412402
X0,Y0 =
78.32

A,B,C = 7.79695368 15.40473984 1.21297074

X0,Y0 =

26.45

A,B,C = 24.72867622 20.22665158 1.007624521

X0,Y0 =

33.11

A,B,C = 1.67223367 18.95024804 1.037336206

X0,Y0 =

4.11

PLACE OBJECTS?

ANSWER 0 OR 1

E:

1
OBJECT ?

L:

37 37
OBJECT ?

E:

ALLOBJECTS[1::]

PLACE OBJECTS?

ANSWER 0 OR 1

E:

1
LOCATION ?

E:

85 70
OBJECT ?

E:

ALLOBJECTS[3::

::]

PLACE OBJECTS?

ANSWER 0 OR 1

E:

0

and the range data picture is



Here detection/recognition is perfect:

ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 1
AT LOCATION 34 42
TO BE = 13

RECOGNISE EXP133
ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 3
AT LOCATION 86 84
TO BE = 39

Note, however, that the credibility of hypothesis object no. 1 is only 13, much lower than 39.

EXPD252. Now

EXPD252+CHNCE1E 4
A,B,C =
11.74561476 26.55120746 2.963593093
D,E,F =
57 20

A,B,C =
27.20373626 6.34450712 6.132045406
D,E,F =
0.3 116

A,B,C =
13.7511964 23.15205734 0.321109472
D,E,F =
10.16

A,B,C =
31.560985 25.4756803 1.617704321
D,E,F =
0.0 123

PLACE OBJECTS?

ANSWER 0 OR 1

E:

1
LOCATION ?

E:

37 37

OBJECT ?

E:

ALLOBJECTS[2;;]

PLACE OBJECTS?

ANSWER 0 OR 1

E:

1

LOCATION ?

E:

95 70

OBJECT ?

E:

ALLOBJECTS[5;;]

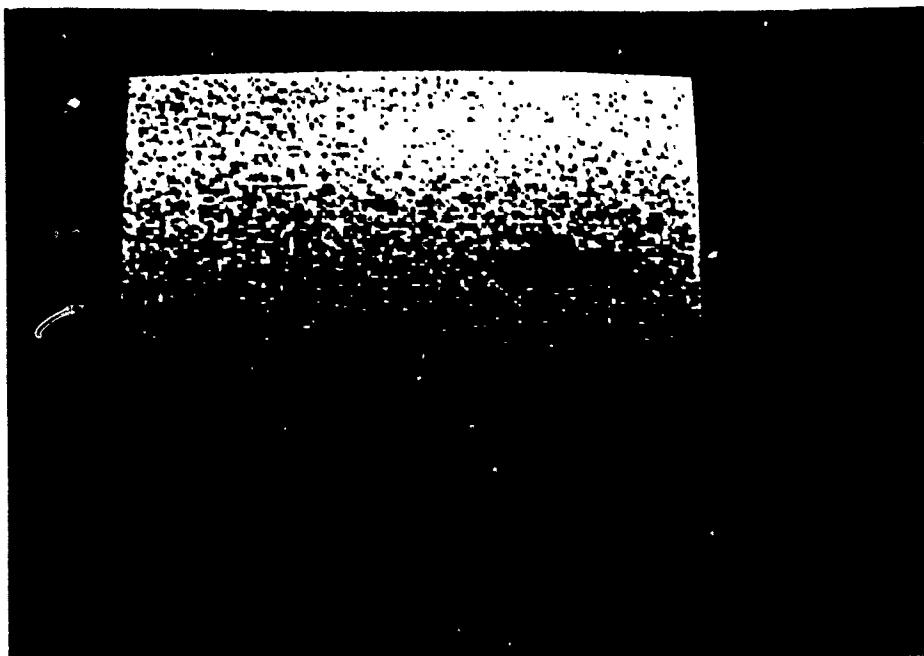
PLACE OBJECTS?

ANSWER 0 OR 1

E:

0

with the appearance



and, again, a correct decision

RECOGNIZE EXP252
ALGORITHM DECIDES CREDIBILITY OF HYPOTHESES:
OBJECT NO. 5
AT LOCATION 92 75
TO BE = 18

ALGORITHM DECIDES CREDIBILITY OF HYPOTHESES:
OBJECT NO. 2
AT LOCATION 34 43
TO BE = 14

Now let us make it harder by putting CMAX ~ 20.

EXPD253M

EXPD253M+GENSCN F 4

A,B,C =
12.73000036 22.64717712 13.10037797
X0,Y0 =
121 106

A,B,C =
5.29573522 19.4249745 1.004025904
X0,Y0 =
126 83

A,B,C =
17.3352531 9.30002932 2.010775642
X0,Y0 =
8 83

A,B,C =
20.53503462 20.3281953 15.93804546
X0,Y0 =
51 69

PLACE OBJECTS?

ANSWER 0 OR 1

E:

1

LOCATION ?

E:

45 45

OBJECT ?

E:

ALLOBJECTS[2;:]

PLACE OBJECTS?

ANSWER 0 OR 1

E:

1

ENTRY ERROR

1

LOCATION ?

E:

85 75

OBJECT ?

E:

ALLOBJECTS[5;:]

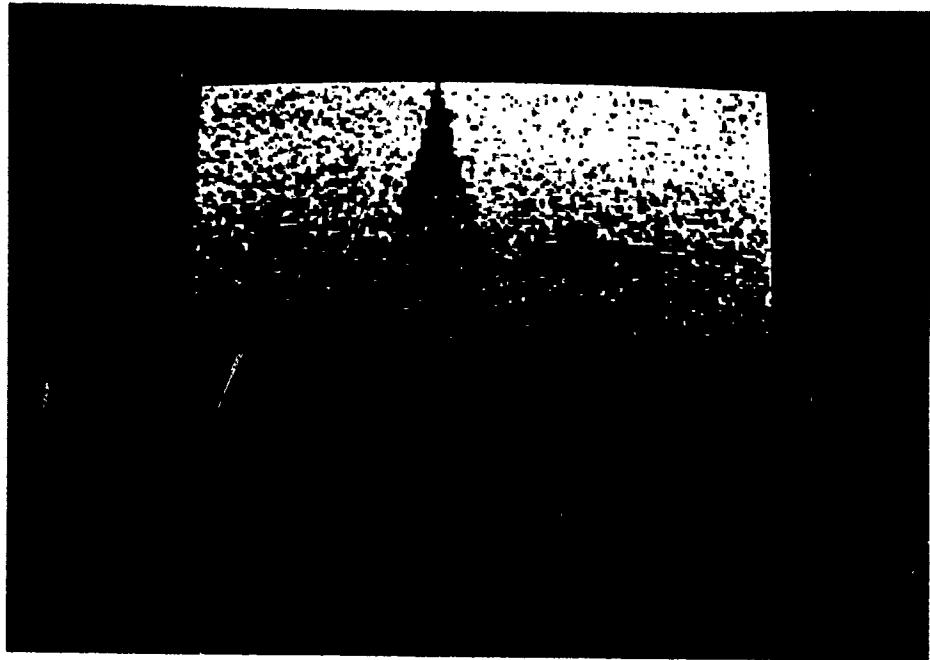
PLACE OBJECTS?

ANSWER 0 OR 1

E:

with the picture

0



The decision is correct:

RECOGNIZE EXPD253M

ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 2
AT LOCATION 47 54
TO BE = 49

ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 5
AT LOCATION 90 82
TO BE = 45

EXPD033M. We now made a cosmetic change in the code. We now print out the credibilities of all object hypothesis above a certain level (NAGREE) and also the final decision.

For

IMPP033..+CHNGC116

11.33827136 45.55635192 2.035422595

X0,Y0 =

0 40

A,B,C =

5.80977654 26.9164962 0.000104102

X0,Y0 =

0 30

A,B,C =

23.66394492 20.41657482 4.022147691

X0,Y0 =

27 100

A,B,C =

23.31575672 7.92729590 7.322177061

X0,Y0 =

48 20

PLACE OBJECTS?

ANSWER 0 OR 1

0:

1

LOCATION ?

0:

45 45

OBJECT ?

0:

ALLOBJECTS[0;;]

PLACE OBJECTS?

ANSWER 0 OR 1

0:

1

LOCATION ?

0:

85 75

OBJECT ?

0:

ALLOBJECTS[3;;]

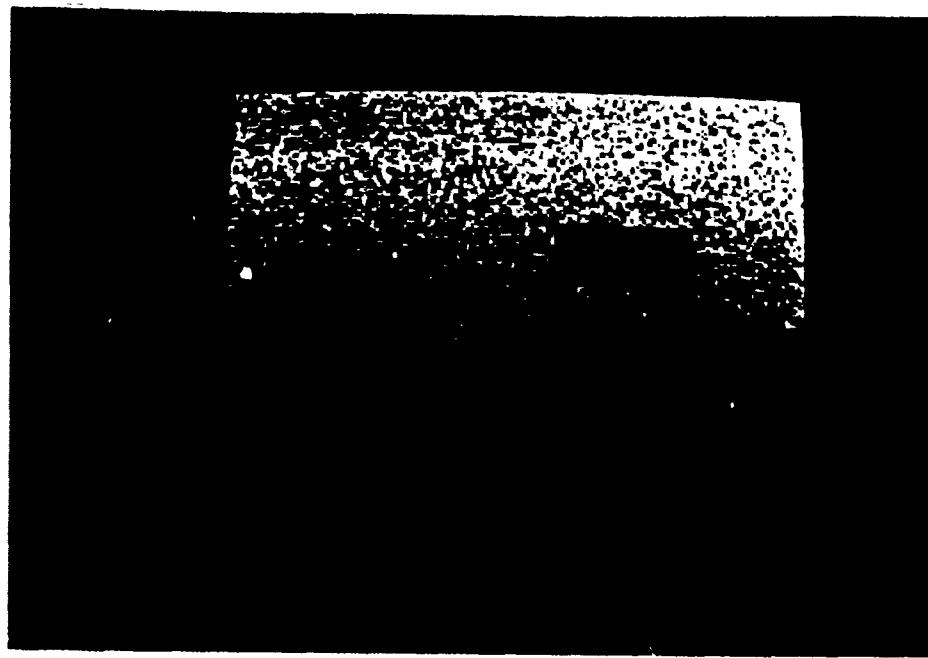
PLACE OBJECTS?

ANSWER 0 OR 1

0:

0

and



the algorithm mistakes object 3 for object 1.

RECOGNIZE EXPRESSION
ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 0
AT LOCATION 86 91
TO DE = 36

ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 4
AT LOCATION 84 90
TO DE = 33

ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 3
AT LOCATION 85 69
TO DE = 32

ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 2
AT LOCATION 86 80
TO DE = 28

ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 5
AT LOCATION 86 91
TO DE = 25

ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 1
AT LOCATION 35 51
TO DE = 13

RESULT OF DECISION FUNCTION IS:
OBJECT NO. 0
AT LOCATION 86 91
WITH CREDIBILITY 36

OBJECT NO. 1
AT LOCATION 35 51
WITH CREDIBILITY 13

To look more closely into this we lowered the threshold constant K from 90% agreement to 85%. We then got the correct decision

RECOGNIZE EXPDCEA^A

ALGORITHM DECIDES CREDIBILITY OF EXPDCEA:

OBJECT NO. 3
AT LOCATION 86 86
TO DE = 59

ALGORITHM DECIDES CREDIBILITY OF PYRONESES:

OBJECT NO. 4
AT LOCATION 88 85
TO DE = 56

ALGORITHM DECIDES CREDIBILITY OF PYRONESES:

OBJECT NO. 0
AT LOCATION 42 51
TO DE = 53

ALGORITHM DECIDES CREDIBILITY OF EXPDCEA:

OBJECT NO. 5
AT LOCATION 40 50
TO DE = 46

ALGORITHM DECIDES CREDIBILITY OF EXPDCEA:

OBJECT NO. 2
AT LOCATION 88 85
TO DE = 42

ALGORITHM DECIDES CREDIBILITY OF EXPDCEA:

OBJECT NO. 1
AT LOCATION 86 51
TO DE = 34

RESULT OF DECISION FUNCTION IF:

OBJECT NO. 3
AT LOCATION 86 86
WITH CREDIBILITY 59

OBJECT NO. 0
AT LOCATION 42 51
WITH CREDIBILITY 53

This turned out to improve performance. The reason is of course that it allows for high noise level. However, it tends to make the lists TESTSEL and TESTSLOC a good deal longer, which requires more CPU time. We shall return to this question in section 12.

EXPD143M We got

EXPD143M<CTVSCEVZ 4
A,P,C =
0.67801412 10.80520010 14.26700001
X0,Y0 =
115 23

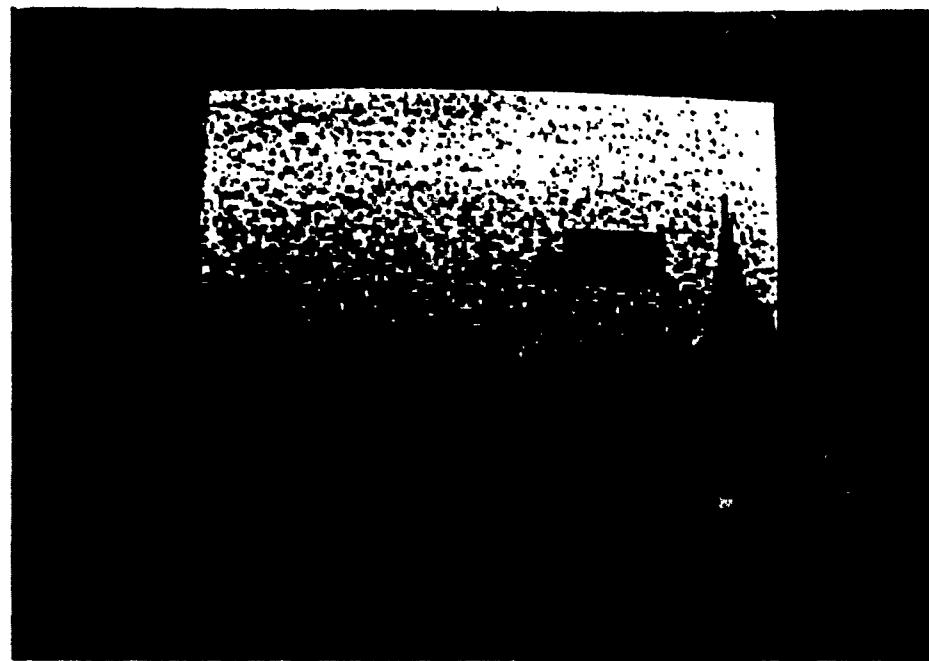
A,P,C =
10.57136000 10.32265332 8.977025041
X0,Y0 =
125 24

A,P,C =
13.34605924 24.63193554 1.275030070
X0,Y0 =
10 30

A,P,C =
3.3128178 4.05413926 19.82570473
X0,Y0 =
57 114

PLACE OBJECTS?
ANSWER 0 OR 1
□:
1
LOCATION ?
□:
45 45
OBJECT ?
□:
ALLOCATES[1;;]
PLACE OBJECTS?
ANSWER 0 OR 1
□:
1
LOCATION ?
□:
85 75
OBJECT ?
□:
ALLOCATES[4;;]
PLACE OBJECTS?
ANSWER 0 OR 1
□:
0

and



The decision was

RECOGNIZE EXPD143H
ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 4
AT LOCATION 87 84
TO BE = 43

ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 3
AT LOCATION 89 84
TO BE = 42

ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 2
AT LOCATION 81 85
TO BE = 35

ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 5
AT LOCATION 84 80
TO BE = 28

ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 0
AT LOCATION 85 78
TO BE = 24

ALGORITHM DECIDES CREDIBILITY OF HYPOTHESIS:
OBJECT NO. 1
AT LOCATION 94 89
TO BE = 13

RESULT OF DECISION FUNCTION IS:
OBJECT NO. 4
AT LOCATION 87 84
WITH CREDIBILITY 43

OBJECT NO. 1
AT LOCATION 94 89
WITH CREDIBILITY 13

which looks correct but is not. Indeed, the object recognized as no. 1 has its center located at the point 94,89 instead of at 45,45. It is just luck that the algorithm claims to have recognized no. 1.

7.4. The experiments carried out so far indicate that for the given model the algorithm

- (i) performs well as a detector/recognizer when there is little or no obscuration
- (ii) performs less well as a recognizer when obscuration occurs, but still fairly well as a detector.

In the latter case lowering the k-value to 85% leads to noticeable better performance.

8. Code

```
    V Z+S1 BOTH S2;N1;N2;N1;N2;VAR1;VAR2
[1]  ACRITERION FOR SMALL IN-VARTANCE
[2]  AND ALSO LARGE OUT-VARTANCE
[3]  N1←pS1
[4]  N2←pS2
[5]  N1←(+/S1)+N1
[6]  N2←(+/S2)+N2
[7]  VAR1←(+/(S1-N1)*2)+N1
[8]  VAR2←(+/(S2-N2)*2)+N2
[9]  Z←(VAR1≤CONST)×VAR2
    V

    V Z+CHANGE FIELD;MEANS;VARS;T
[1]  T←0
[2]  MEANS←FIELD
[3]  VARS←FIELD*2
[4]  LOOP:MEANS←MEANS+NJ[T;0]Φ[0] NJ[T;1]ΦFIELD
[5]  VARS←VARS+NJ[T;0]Φ[0] NJ[T;1]ΦFIELD*2
[6]  T←T+1
[7]  +(T<8)/LOOP
[8]  Z←(VARS-(MEANS*2)+9)<12×TCONST
    V

    V Z+FEW CLEAN X;T;NNEIGH
[1]  REMOVES POINTS WITH FEW NEIGHBORS FROM
[2]  ABINARY IMAGE X
[3]  T←0
[4]  NNEIGH←(pX)p0
[5]  LOOP:NNEIGH←NNEIGH+NJ[T;0]Φ[0] NJ[T;1]ΦX
[6]  T←T+1
[7]  +(T<8)/LOOP
[8]  Z←X^NNEIGH>FEW
    V

    V COLLECT;T;C;INTERIOR
[1]  ACOLLECTS ALL OBJECT OBJECTTYPES INTO 3-DIM ARRAY
[2]  ACALLED ALLOBJECTS
[3]  EACH OBJECT REPRESENTED AS L×L DIGITAL BINARY PICTURE
[4]  T←0
[5]  ALLOBJECTS←(NOBJECTS,L,L)p0
[6]  LOOP:C←('OBJECT').YT
[7]  C←C×LSCALE
[8]  C[;0]←C[;0]+0.51
[9]  C[;1]←C[;1]++19
[10]  INSIDE
[11]  ALLOBJECTS[T;;]←INTERIOR
[12]  T←T+1
[13]  +(T<NOBJECTS)/LOOP
    V
```

```
    V COORDS=COORD BOOT.E;ROWS;ROWNUMS;COLS;COLNUMS;L1;L2
[1]  ATRANSFORMS L1×L2 BOOT.EAN MATRTX BOOT.E INTO 2-COL.UNN MATRTX COORDS
[2]  ACONTAINING X.Y-COORDINATES
[3]  L1+(pBOOT.E)[0]
[4]  L2+(pBOOT.E)[1]
[5]  NROWS++/+/BOOT.E
[6]  COORDS+(NROWS.2)p0
[7]  ROWS+(,L1)•.×L2p1
[8]  ROWNUMS=ROWS×BOOT.E
[9]  COORDS[;0]+(.BOOT.E)/,ROWNUMS
[10]  COLS+(L1p1)•.×L2
[11]  COLNUMS=COLS×BOOT.E
[12]  COORDS[;1]+(.BOOT.E)/,COLNUMS
    V

    V Z=S1 CORR S2
[1]  ACOMPUTES CORRELATION
[2]  Z+(+/+/S1×S2)++/+/(~S1)×~S2
    V

    V Z+K CUTOFF FIELD;NZ;Q1;Q2;Q3
[1]  ACOMPUTES THRESHOLD VALUE FOR FTELD
[2]  Z+.FTELD
[3]  Z+Z[ΔZ]
[4]  NZ+pZ
[5]  Q1+Z[10.25×NZ]
[6]  Q2+Z[10.5×NZ]
[7]  Q3+Z[10.75×NZ]
[8]  Z+Q2+K×(Q3-Q1)+2
    V

    V DISPLAY Z;MIN;MAX;D
[1]  MAX+[1/Z
[2]  MIN+[1/Z
[3]  D+MAX-MIN
[4]  φ[0]q(' ..o*□')+/((Z-MIN)+D)•.≥ 0 0.2 0.4 0.6 0.8]
    V

    V Z=ELDIST;I;J
[1]  ACOMPUTES HAMMING DISTANCE MATRTX Z
[2]  AFOR ALL SHAPE ELEMENTS
[3]  Z+(2pNSLICRS)p0
[4]  I+1+J+0
[5]  LOOP:Z[I;J]+Z[J;I]++/+/|SLICRS[I;:]-SLICRS[J;:|
[6]  I+I+1
[7]  +(I<NSLICRS)/LOOP
[8]  I+1+J+J+1
[9]  +(J<NSLICRS-1)/LOOP
    V
```

V ESTGENS FIELD;T;TEMP
[1] @ESTIMATES GENERATORS FROM DATA=FILE.D
[2] T+0
[3] RESULT+ 0 3 p0
[4] @RESULT HAS AS ROWS U,V,T-VALUES
[5] LOOP:TEMP+SLICES[T;:]
[6] LTEMPX+(pTEMP)[0]
[7] LTEMPY+(pTEMP)[1]
[8] VIEW FIELD
[9] SEARCH CRITERION
[10] RESULT+RESULT.[0] T,COORDS
[11] T+T+1
[12] +(T<(pGENERATORS)[0])/LRS:
 V

 V Z+HLIST EVIDENCE TLIST
[1] @COMPUTES DEGREE OF EVIDENCE FOR HYPOTHESES (OBJECT)
[2] @SPECIFIED BY SHAPE ELEMENTS\LOCATIONS IN 3-COLUMN MATRIX HLIST
[3] @SUPPORTED BY 3-COLUMN MATRIX TLIST OF TEST RESULTS
[4] @GLOBAL SCALAR WEIGHT EXPRESSES IMPORTANCE OF DISTANCE
[5] @BETWEEN SHAPE ELEMENTS AS COMPARED TO EUCLIDEAN DISTANCE
[6] @BETWEEN LOCATIONS
[7] Z+DISTEL[HLIST[;0];TLIST[;0]]
[8] Z+-(Z*WEIGHT)++/[1](0 1 2 1 &HLIST[; 1 2]..-TLIST[; 1 2])*2
 V

 V Z+FINDGENS;T;ALLOBJECTS1
[1] @COMPUTES GENERATORS (SHAPE ELEMENTS) FROM OBJECTS STORED
[2] @IN ARRAY WITH GLOBAL NAME ALLOBJECTS
[3] @RESULT STORED IN 3-DIM ARRAY Z
[4] Z+(0,LTEMPX,LTEMPY)p0
[5] T+0
[6] ALLOBJECTS1+((pALLOBJECTS)+(0,(2*LTEMPX),(2*LTEMPY)))p0
[7] ALLOBJECTS1[;LTEMPX+1L;LTEMPY+1L]+ALLOBJECTS
[8] LOOP:IMAGE+ALLOBJECTS1[T;:]
[9] Z+Z.[0] DIST SELECT IMAGE
[10] T+T+1
[11] +(T<NOBJECTS)/LOOP ~
[12] NG+(pZ)[0]
 V

 V Z+S1 FISHER S2;N1;N2;M1;M2;VAR1;VAR2
[1] @COMPUTES FISHER TYPE CRITERION FOR VARIANCES
[2] @OF THE TWO SAMPLES S1 AND S2
[3] N1+pS1
[4] N2+pS2
[5] M1+(+/S1)+N1
[6] M2+(+/S2)+N2
[7] VAR1+(/(S1-M1)*2)+N1
[8] VAR2+(/(S2-M2)*2)+N2
[9] Z+(-(•VAR1+0.0001)-•VAR2+0.0001)+((+N1)++N2)*0.5
 V

```
    V Z←FORMLIST HYP;T
[1]  ⋀FORMS LISTS HPLIST, ETC FOR HYPOTHESES NO. HYP
[2]  NT←(pTESTSEL)[0]
[3]  TLIST←(NT,3)ρ0
[4]  TLIST[;0]←TESTSEL
[5]  TLIST[; 1 2]←TESTSLOC
[6]  ↳'NH←(pHYPEL',(vHYP).')'0]'
[7]  HLIST←(NH,3)ρ0
[8]  ↳'HLIST[;0]←HYPEL',(vHYP)
[9]  ↳'HLIST[;1 2]←HYPLOC',vHYP
    V

    V HLIST FULLTEST TLIST;I;J;NH;TLIST1
[1]  ⋀CARRIES OUT ALL TESTS FOR GIVEN LISTS
[2]  ⋀TRYING TO MATCH RELATIVE POSITIONS
[3]  I←J←0
[4]  NH←(pHLIST)[0]
[5]  NT←(pTLIST)[0]
[6]  NAGREEMAX←0
[7]  KEEPMAX←1NT
[8]  LOOP:OFFSET←HLIST[I; 1 2]-TLIST[J; 1 2]
[9]  TLIST1←TLIST
[10]  TLIST1[; 1 2]←TLIST[; 1 2]+(NTρ1).. $\times$ OFFSET
[11]  KEEPM←HLIST MATCH TLIST1
[12]  →(NAGREEMENTS≤NAGREEMAX)/LOOP2
[13]  KEEPMAX←KEEP
[14]  NAGREEMAX←NAGREEMENTS
[15]  LOOP2:I←I+1
[16]  +(I<NH)/LOOP
[17]  I←0
[18]  J←J+1
[19]  +(J<NT)/LOOP
    V

    V GENHYPPLISTS;T;S;OBJECT;SUR;HAMMING;BEST
[1]  ⋀COMPUTES LIST CALLED HYPEL OF SHAPE ELEMENTS FOR EACH OBJECT
[2]  ⋀AND LIST CALLED HYPLOC OF CENTER COORDINATES OF FITTED SHAPE ELEMENTS
[3]  ⋀ASSUMES AVAILABLE:3-DTM ARRAY SLICES OF
[4]  ⋀SHAPE ELEMENTS
[5]  NSLICES←(pSLICES)[0]
[6]  T←0
[7]  OBJECT←(L+2×LTEMPX.LTEMPY)ρ0
[8]  LOOP1:OBJECT[LTEMPX+1L;LTEMPY+1L]←ALLOBJECTS[T;:]
[9]  ⋀NOW COMPUTE LIST OF SURPICTURES ALONG BOUNDARY OF OBJECT
[10] ⋀SEPARATED BY DISTANCE CALLED DISTHYP
[11]  SUB←DTSTHYP SELECT OBJECT
[12]  ↳('HYPLOC',vT).'+CENTERS'
[13]  S←0
[14]  ↳('HYPEL',vT).'+NBρ0'
[15]  LOOP2:HAMMING++/+/[SLICES-(NSLICESρ1).. $\times$ SUR[S;:]
[16]  BEST←HAMMING1L/HAMMING
[17]  ↳('HYPEL',vT).'[S]←BEST'
[18]  S←S+1
[19]  +(S<NB)/LOOP2
[20]  T←T+1
[21]  +(T<NOBJECTS)/LOOP1
    V
```

▼ Z←GENOBJECT NPOINTS;T
[1] ⌈GENERATES POLYGON WITH NPOINTS VERTICES
[2] Z←(NPOINTS,2)◊0
[3] 'X-COORDINATES'
[4] Z[;0]+◊
[5] 'Y-COORDINATES'
[6] Z[;1]+◊
▼

▼ RANGE←GENSCENE NHILLS;X;ANS;LOCATION;OBJECT
[1] ⌈GENERATES SCENERY WITH NHILLS HILLS,
[2] ⌈SKY. AND OBJECTS
[3] Z←GENTERRAIN NHILLS
[4] Z←ZSCALE×Z
[5] Z[:(LY-LSKY)+1]←SKY
[6] ⌈ZSCALE IS HEIGHT SCALE
[7] Z←Z+WHITE SMALLSCALE
[8] X+0
[9] RANGE←(LX,LY)◊1000
[10] LOOP1:RANGE[X;]+STGHT1 Z[X;]
[11] X+X+1
[12] →(X<LY)/LOOP1
[13] RANGE←RANGE+10.5+WHITE BACKGROUNDERROR
[14] LOOP2:'PLACE OBJECTS?'
[15] 'ANSWER 0 OR 1'
[16] ANS+◊
[17] →(ANS=0)/LOOP3
[18] 'LOCATION ?'
[19] LOCATION+◊
[20] 'OBJECT ?'
[21] OBJECT+◊
[22] LOCATION PLACE OBJECT
[23] →LOOP2
[24] LOOP3:RANGE←RANGE+10.5+WHITE OBJECTERROR
▼

▽ TERRAIN+GENTERRAIN NHILLS;T;A;B;C;XS;YS;X0;Y0
[1] □ GENERATES BACKGROUND OF HILLY LANDSCAPE
[2] □ WITH NHILLS HILLS
[3] □ SHAPE OF HILL CONTROLLED BY A'S AND B'S
[4] □ HEIGHT OF HILLS CONTROLLED BY C'S
[5] □ GLOBAL VARIABLE POWER CONTROLS POWER LAW
[6] □ DISTRIBUTION FOR HILL HEIGHTS; NEGATIVE VALUES GIVE PARETO
[7] □ DISTRIBUTIONS
[8] T+0
[9] TERRAIN+(LX.LY)ρ0
[10] XS+1LX
[11] YS+1LY
[12] LOOP:A+AMIN+(AMAX-AMIN)×1E-8×?1000000000
[13] B+BMIN+(BMAX-BMIN)×1E-8×?1000000000
[14] C+CMIN+(CMAX-CMIN)×(1E-8×?1000000000)*POWER
[15] 'A,B,C ='
[16] A,B,C
[17] X0+?LX
[18] Y0+?LY
[19] 'X0,Y0 ='
[20] X0,Y0
[21] 10
[22] TERRAIN+TERRAIN+C+1+(((XS-X0)+A)*2)+((YS-Y0)+B)*2)*0.5
[23] T+T+1
[24] +(T<NHILLS)/LOOP
▽

▽ Z+IBOUNDARY IMAGE;T;OUT
[1] □ COMPUTES INNER BOUNDARY OF SET IMAGE
[2] □ USES NEIGHBORHOOD DEFINITION IN TERMS OF TWO COLUMN MATRIX J
[3] □ BOUNDARY OF MATRIX IMAGE MUST BE FREE OF 1'S
[4] □ NJ+ 8 2 0 -1 1 -1 1 0 1 1 0 1 -1 1 -1 0 -1 -1
[5] OUT+(ρIMAGE)ρ0
[6] T+0
[7] LOOP:OUT+OUTv~NJ[T;0]Φ[0] NJ[T;1]ΦIMAGE
[8] T+T+1
[9] +(T<(ρNJ)[0]))/LOOP
[10] Z+IMAGE^OUT
▽

V INSIDE;V;C1;Y;Z;X;NX;LEFT;RIGHT;IN
[1] A COMPUTES DIGITAL PICTURE OF POINTS IN
[2] A $L \times L$ -LATTICE INSIDE POLYGON
[3] A DESCRIBED BY $L \times 2$ MATRTX C
[4] V+(1L)+L
[5] C1←(1Φ[0] C)-C
[6] INTERIOR←(L,L)ρ0
[7] +((1E-8>|C1[;1]|)∧0=1|L×C[;1])/WARNING
[8] Y+0
[9] LOOP1:Z+(Y-C[;1])+C1[;1]+0=C1[;1]
[10] X+C[;0]+Z×C1[;0]
[11] Z1+Z+1
[12] X+((0≠C1[;1])∧(Z1≥1)∧(Z1≤2)∧(Z1<2)∨(Z1=2)∧0>C1[;1]×1Φ[0] C1[;1])/X
[13] X+X[AX]
[14] NX+ρX
[15] +(0=NX)/LOOP2
[16] LEFT←X[2×1NX+2]
[17] RIGHT←X[1+2×1NX+2]
[18] IN←(V/(V.≥LEFT)∧V.≤RIGHT)/V
[19] INTERIOR[L×IN;L×Y]←1
[20] LOOP2:Y+Y+L
[21] +(Y<1)/LOOP1
[22] +0
[23] WARNING:'DATA NOT WELL CONDITIONED FOR ALGORITHM! '
 V

 V Z+S1 KOLMOGOROV S2;N1;N2;ZS;F1;F2
[1] A COMPUTES KOLMOGOROV-SMIRNOV
[2] A STATISTIC (STANDARDIZED) FOR SAMPLES S1 AND S2
[3] N1←ρS1
[4] N2←ρS2
[5] ZS←S1,S2
[6] ZS←ZS[AZS]
[7] F1←(+/ZS.≥S1)+N1
[8] F2←(+/ZS.≥S2)+N2
[9] Z←(1/(F1-F2)×(N1×N2+N1+N2))×0.5
 V

V Z+K LEBESGUE SLICES;T;NSTNBALL;S;MAX;CLOSE
[1] ESTIMATES LEBESGUE SET BY CHOOSING K
[2] BALLS OF RADIUS=RADIUS WITH HIGHEST DENSITY
[3] Z+(0,LTEMPX,LTEMPY),0
[4] T+0
[5] LOOP1:+(0=(pSLICES)[0])/END
[6] S+0
[7] NSTNBALL+(pSLICES)[0],0
[8] LOOP2:NSINBALL[S]+SLICES[S,:]; NSTNBALL SLICES
[9] S+S+1
[10] +(S<(pSLICES)[0])/LOOP2
[11] MAX+NSINBALL,[/NSTNBALL
[12] Z+Z,[0] SLICES[MAX,:]
[13] CLOSE+RADIUSz+/-|SLICES-((pSLICES)[0],0),0..xSLICES[MAX,:]
[14] SLICES+(-CLOSE)/[0] SLICES
[15] T+T+1
[16] +(T<K)/LOOP1
[17] NOW DELETE EXTREME GENERATOR ESTIMATORS
[18] END:DROP+(0.3xLTEMPXxLTEMPY)<|(0.5xLTEMPXxLTEMPY)-+/-Z
[19] Z+(-DROP)/[0] Z
[20] NSLICES+(pZ)[0]
 V

 V KEEP+HYPLIST MATCH TLIST;AGREEMENTS;NROWS;NCOLS
[1] TRIES TO MATCH SUBSET OF TEST RESULTS IN 3-COLUMN
[2] MATRIX TLIST WITH SUBSET OF SHAPE ELEMENTS/LOCATIONS
[3] A
[4] RESULT Z IS VECTOR OF SUBSCRIPTS IN TLIST THAT SHOULD BE
[5] DROPPED FROM TLIST AFTER MATCHING
[6] AGREEMENTS+HYPLIST EVIDENCE TLIST
[7] NROWS+(pAGREEMENTS)[0]
[8] NCOLS+(pAGREEMENTS)[1]
[9] KEEP+1NCOLS
[10] MAX+1/[0] AGREEMENTS
[11] AGREEMENTS+(AGREEMENTS=(NROWS,0),0..xMAX)^AGREEMENTS>AGREELEVEL
[12] NAGREEMENTS+/-1/[0] AGREEMENTS
[13] +(NAGREE>NAGREEMENTS)/0
[14] KEEP+(-1/[0] AGREEMENTS)/1NCOLS
 V

 V Z+S1 MAXDEV S2
[1] Z+((1/.S2)-1/.S2)+0.0001+(1/.S1)-1/.S1
 V

 V CENTER+MAXLOC FIELD;BOOLE;U;V
[1] COMPUTES LOCATION OF SOME (!) MAXIMUM OF FIELD
[2] CENTER+2,0
[3] BOOLE+FIELD=1/[1/FIELD
[4] U+(-V/BOOLE),1
[5] V+(-BOOLE[U,]),1
[6] CENTER[0]+(UxSTEP)+LTEMPX+2
[7] CENTER[1]+(VxSTEP)+LTEMPY+2
 V

▼ Z+CENTER NINBALL SLICRS
[1] A3-DIM ARRAY CONTAINS SLICRS OF SIZE LTEMPX×LTEMPY
[2] ACOMPUTES NUMBER OF SLICRS IN BALL CENTERED SLICR= CENTER
[3] ARADIUS OF BALL=GLOBAL VARTABLE RADTUS
[4] Z++/RADIUS>+/+/|SLICRS-((pSLICRS)[0]p1).. \times CENTER
 ▼

 ▼ LOCATION PLACE OBJECT
[1] M+RANGE[(-(L+2)+(1L)+LOCATION[0];(1L)+LOCATION[1]]
[2] RANGE[(-(L+2)+(1L)+LOCATION[0];(1L)+LOCATION[1]) \leftarrow ML(LOCATION[1]
 \times OBJECT)+10000 \times 1-OBJECT
 ▼

 ▼ RECOGNIZE IMAGE;S;T;LOCATIONS;NAGREEMAXS;PERM;LOCLIST;NEWLOC
[1] ACOMPUTES CREDIBILITY OF LIKELY HYPOTHESES
[2] FOR RANGE DATA IMAGE
[3] T \leftarrow 0
[4] TESTS1 IMAGE
[5] LOCATIONS \leftarrow 0 2 p0
[6] NAGREEMAXS \leftarrow 10
[7] NT \leftarrow (pTESTSEL)[0]
[8] \rightarrow (0=NT)/0
[9] LOOP: x'FORMLIST', ∇ T
[10] HLIST FULLTEST TLIST
[11] \rightarrow (NT=pKEEPMAX)/LOOP1
[12] LOST \leftarrow (~(1NT) ϵ KEEPMAX)/1NT
[13] LOCATIONS \leftarrow LOCATIONS,[0](+/[0] TLIST[LOST; 1 2]) \leftarrow pLOST
[14] NAGREEMAXS \leftarrow NAGREEMAXS.NAGREEMAX
[15] LOOP1:T \leftarrow T+1
[16] \rightarrow (T<NOBJECTS)/LOOP
[17] S \leftarrow 0
[18] PERM \leftarrow 1NAGREEMAXS
[19] LOOP2: \rightarrow (S=pNAGREEMAXS)/SUMMARY
[20] 'ALGORITHM DECIDES CREDIBILITY OF HYPOTHESES:'
[21] 'OBJECT NO. '. ∇ ((1pNAGREEMAXS)[PRRM])[S]
[22] 'AT LOCATION '. ∇ (LOCATIONS[PRRM;])[S;]
[23] 'TO BE = '. ∇ (NAGREEMAXS[PRRM])[S]
[24] 10
[25] S \leftarrow S+1
[26] \rightarrow LOOP2
[27] SUMMARY:10
[28] 'RESULT OF DECISION FUNCTION IS:'
[29] S \leftarrow 0
[30] \rightarrow (S=pNAGREEMAXS)/0
[31] LOCLIST \leftarrow 1 2 p10000000
[32] LOOP3:NEWLOC \leftarrow (LOCATIONS[PRRM;])[S;]
[33] \rightarrow (MINDIST>1/+/|LOCLIST-((pLOCLIST)[0]p1).. \times NRWLOC)/LOOP4
[34] LOCLIST \leftarrow LOCLIST,[0] NEWLOC
[35] 'OBJECT NO. '. ∇ ((1pNAGREEMAXS)[PRRM])[S]
[36] 'AT LOCATION '. ∇ NEWLOC
[37] 'WITH CREDIBILITY '. ∇ (NAGREEMAXS[PRRM])[S]
[38] 10
[39] LOOP4:S \leftarrow S+1
[40] \rightarrow (S<pNAGREEMAXS)/LOOP3
 ▼

V RECOGNIZER;T;NT;LOST;CRED
[1] ARECOGNITION ALGORITHM FOR RECOGNIZING OBJECTS
[2] IN IMAGE REPRESENTED BY GLOBAL VECTOR CALLED TLIST
[3] T+0
[4] LOOP:#'FORMLIST ',#T
[5] HLIST FULLTEST TLIST
[6] →(0=NT)/0
[7] →(NT=KEEPMAX)/LOOP1
[8] 'ALGORITHM SUGGESTS HYPOTHESES: OBJECT NO. ',#T,' IN IMAGE'
[9] LOST→(~(,NT)≤KEEPMAX)/,NT
[10] LOCATION→L(+/[0] TLIST[LOST; 1 2])+,LOST
[11] 'WITH LOCATION AT ',#LOCATION
[12] 'WITH CREDIBILITY ',#NAGREEMAX
[13] ,0
[14] LOOP1:T+T+1
[15] →(T<NOBJECTS)/LOOP
 V

 V SCALED+LEVELS SCALE IMAGE;MIN;MAX
[1] SCALE IMAGE TO GREY LEVELS
[2] LEFT ARGUMENT =NO. OF GREY LEVELS
[3] IMAGE→0[IMAGE]
[4] MIN→L/L/IMAGE
[5] MAX→Γ/Γ/IMAGE
[6] SCALED→L(LEVELS-1)×(IMAGE-MIN)+MAX-MIN
 V

 V Z+SCALE SCALING OBJECT
[1] SCALE AND PLACES OBJECT IN L×L ARRAY
[2] C→SCALE×OBJECT
[3] C[;0]+C[;0]+#2
[4] INSTDE
[5] Z→INTERIOR
[6] Z→Φ[0]#Z
 V

 V Z+DIST SCREEN BOUND;CLOSE
[1] SELECTS SUBSET OF POINTS FROM BOUND=2COLUMN MATRIX
[2] PTS SEPARATED BY DISTANCE ≥DIST
[3] Z→ 0 2 #0
[4] LOOP:→(0=(,BOUND)[0])/0
[5] Z→Z.[0] BOUND[0;]
[6] CLOSE→DIST≥+/, BOUND-((,BOUND)[0],p1)×,×BOUND[0;]
[7] BOUND→(~CLOSE)/[0] BOUND
[8] →LOOP
 V

```
    V SEARCH FIELD;FIELD1;LXF;LYF;THRESHOLD;BOOLR;XS;YS;Z
[1]  A COMPUTES LIST COORDS OF SEPARATED MAXIMA OF FIELD
[2]  A MIN SEPARATION DISTANCE=NEIGH
[3]  A CONSTANT K DECIDES CUTOFF LEVEL
[4]  LXF+(pFIELD)[0]
[5]  LYF+(pFIELD)[1]
[6]  COORDS+ 0 2 p0
[7]  THRESHOLD+K×LTEMPX×LTEMPY
[8]  FIELD1+FIELD
[9]  LOOP:Z+2p0
[10]  BOOLR+FIELD1=1/1/FIELD1
[11]  Z[0]+(v/BOOLR),1
[12]  Z[1]+(.BOOLR[Z[0]];),1
[13]  +(FIELD1[Z[0];Z[1]]<THRESHOLD)/0
[14]  COORDS+COORDS,[0] Z
[15]  XS+(0[Z[0]-NEIGH].LXF[Z[0]]+NEIGH
[16]  YS+(0[Z[1]-NEIGH].LYF[Z[1]]+NEIGH
[17]  FIELD1[XS[0]+,XS[1]-XS[0];YS[0]+,YS[1]-YS[0]]+-100000000
[18]  +LOOP
    V

    V SEE IMAGE
[1]  ('□')[Φ[0]&IMAGE]
    V

    V SEEGENS;T;FRAME
[1]  T+0
[2]  FRAME+(2+LTEMPX.LTEMPY)p'*'
[3]  LOOP:FRAME[1+LTEMPX;1+LTEMPY]+('□')[Φ[0]&SLICES[T;;]]
[4]  'SHAPE ELEMENT NO. ',T
[5]  FRAME
[6]  ,0
[7]  T+T+1
[8]  +(T<NSLICES)/LOOP
    V

    V Z+DIST SELECT IMAGE;BOUNDARY;NEWBOUNDARY;T;XVEC;YVEC
[1]  A SELECTS SURPICTURES OF SIZE LTEMPX×LTEMPY FROM IMAGE
[2]  A CENTERED AT POINTS ALONG BOUNDARY OF IMAGE
[3]  A SEPARATED BY DISTANCE DIST
[4]  A WARNING: SRT IN IMAGE MUST BE AT LEAST LTEMPX.LTEMPY FROM
[5]  A BORDER OF BOOLEAN MATRIX IMAGE
[6]  BOUNDARY+IBOUNDARY IMAGE
[7]  NEWBOUNDARY+DIST SCREEN COORD BOUNDARY
[8]  NB+(pNEWBOUNDARY)[0]
[9]  CENTERS+(NB.2)p0
[10]  Z+(NB.LTEMPX.LTEMPY)p0
[11]  XVEC+(-LTEMPX+2)+,LTEMPX
[12]  YVEC+(-LTEMPY+2)+,LTEMPY
[13]  T+0
[14]  LOOP:Z[T;;]+IMAGE[NEWBOUNDARY[T;0]+XVEC;NEWBOUNDARY[T;1]+YVEC]
[15]  CENTERS[T;]+NEWBOUNDARY[T;]
[16]  T+T+1
[17]  +(T<NB)/LOOP
    V
```

V RESULT+SELECTALL;T;ALLOBJECTS1
[1] @ITERATES FUNCTION CALLED SELECT ON ALL OBJECTS
[2] T+0
[3] RESULT+(0,LTEMPX,LTEMPY)P0
[4] ALLOBJECTS1+(NOBJECTS,(L+2×LTEMPX).L+2×LTEMPY)P0
[5] ALLOBJECTS1[;LTEMPX+1L;LTEMPY+1L]←ALLOBJECTS
[6] LOOP:RESULT+RESULT,[0] DTST SELECT ALLOBJECTS1[T;:]
[7] T+T+1
[8] +(T<NOBJECTS)/LOOP
 V

 V Z←SIGHT1 FIELD;APPEAR;T
[1] @COMPUTES RANGE DATA FROM GIVEN HEIGHT FIELD
[2] @ELEVATION ANGLR IS PSI IN RADTANS
[3] APPEAR+FIELD+(LY)×30PSI
[4] APPEAR+/\APPEAR
[5] Z+LYP0
[6] T+1
[7] LOOP:Z[T]+(APPEAR[T]>/\APPEAR[1T])×T
[8] Z[T]+Z[T]+Z[T-1]×APPEAR[T]≤/\APPEAR[1T]
[9] T+T+1
[10] +(T<LY)/LOOP
 V

 V Z←SKY
[1] @GENERATES BACKGROUND SKY
[2] Z+(LX.LSKY)PDMAX
 V

 V Z←S1 STUDENT S2;N1;N2;M1;M2;SD1;SD2
[1] @STUDENT'S CRITERION FOR TWO SAMPLES S1 AND S2
[2] N1+P1
[3] N2+P2
[4] M1+(/S1)+N1
[5] M2+(/S2)+N2
[6] SD1+((/(S1-M1)*2)+N1)*0.5
[7] SD2+((/(S2-M2)*2)+N2)*0.5
[8] Z+((M1-M2)+0.0001+((SD1*2)+N1)+(SD2*2)+N2)*0.5
 V

 V TESTS IMAGE;NSLICES;T
[1] @COMPUTES VECTOR TESTSET OF SHAPE ELEMENTS
[2] @DETECTED IN IMAGE. AND 2-COLUMN MATRIX TESTSLOC
[3] @WITH THE COORDINATES OF THEIR CENTERS
[4] @USES 3-DIM ARRAY SLICES OF SELECTED GENERATORS=
[5] @SHAPE ELEMENTS
[6] TESTSEL+10
[7] TESTSLOC← 0 2 P0
[8] NSLICES+(P1SLICES)[0]
[9] T+0
[10] LOOP1:TEMP+SLICES[T;:]
[11] VIEW1 CHANGE IMAGE
[12] SEARCH CRITERION
[13] +(0=(P1COORDS)[0])/LOOP2
[14] TESTSEL+TESTSEL.((P1COORDS)[0])P1T
[15] TESTSLOC+TESTSLOC.[0] TRANSFUVS COORDS
[16] LOOP2:T+T+1
[17] +(T<NSLICES)/LOOP1

▼ TRSTS1 IMAGE;T;S;BOUNDARY;NBOUND;BOOL;MX;MY;THRESHOLD
[1] COMPUTES VECTOR TESTSEL OF SHAPE ELEMENTS
[2] DETECTED IN IMAGE. AND 2-COL MATRIX TESTSLOC
[3] WITH THE COORDINATES OF THEIR CENTERS
[4] USES 3-DIM ARRAY SLICES OF SELECTED GENERATORS=

 SHAPE ELEMENTS
[6] AMUCH REDUCED SEARCH EFFORT

 TESTSEL \leftarrow 0
[8] TESTSLOC \leftarrow 0 2 p0
[9] S \leftarrow T \leftarrow 0
[10] IMAGE \leftarrow FEW CLEAN CHANGE IMAGE
[11] BOUNDARY \leftarrow COORD IBOUNDARY \sim IMAGE
[12] MX \leftarrow LLTEMPX \leftarrow 2
[13] MY \leftarrow LLTEMPY \leftarrow 2
[14] THRESHOLD \leftarrow K \times LLTEMPX \times LLTEMPY
[15] BOOLE \leftarrow ((-MX) \times BOUNDARY[;0]) \wedge BOUNDARY[;0] \times LX \times MX
[16] BOOLE \leftarrow BOOLE \wedge ((-MY) \times BOUNDARY[;1]) \wedge BOUNDARY[;1] \times LY \times MY
[17] BOUNDARY \leftarrow BOOLE/[0] BOUNDARY
[18] NBOUND \leftarrow (pBOUNDARY)[0]
[19] LOOP1: TEMP \leftarrow SLICES[T;;]
[20] \rightarrow (THRESHOLD \times TEMP CORR IMAGE[MX+BOUNDARY[S;0]+1LLTEMPX;
 MY+BOUNDARY[S;1]+1LLTEMPY])/LOOP2
[21] TESTSEL \leftarrow TESTSEL.T
[22] TESTSLOC \leftarrow TESTSLOC.[0] BOUNDARY[S;]
[23] LOOP2: S \leftarrow S+1
[24] \rightarrow (S \times NBOUND)/LOOP1
[25] S \leftarrow 0
[26] T \leftarrow T+1
[27] \rightarrow (T \times NSLICES)/LOOP1
 ▼

 ▼ Z \leftarrow TRANSFUVS UVS
[1] TRANSFORMS UV-COORDINATES TO XY'S
[2] Z \leftarrow (pUVS)p0
[3] Z[;0] \leftarrow (UVS[;0] \times STEP)+LLTEMPX \leftarrow 2
[4] Z[;1] \leftarrow (UVS[;1] \times STEP)+LLTEMPY \leftarrow 2
 ▼

 ▼ Z \leftarrow TURN MATRIX
[1] TURNS MATRTX FROM MATRIX COORDINATES TO XY'S
[2] Z \leftarrow Q[0]QMATRIX
 ▼

```
    V VIEW OBS;NUMX;NUMY;U;V;SAMPLEIN;SAMPLEOUT
[1]  A COMPUTES CRITERION FOR RANGE DATA
[2]  A USES CRITERION DEFINED BY 'TEST' V
[3]    NUMX+L(LX-LTEMPX)+STEP
[4]    NUMY+L(LY-LTEMPY)+STEP
[5]    CRITERION+(NUMX.NUMY),0
[6]    U+V+0
[7]  LOOP1: SAMPLE+OBS[(U*STEP)+,LTEMPX;(V*STEP)+,LTEMPY]
[8]    SAMPLEIN+(.TEMP)/.SAMPLE
[9]    SAMPLEOUT+(~.TEMP)/.SAMPLE
[10]   CRITERION[U;V]+2('SAMPLEIN ','TEST'), 'SAMPLEOUT'
[11]   U+U+1
[12]   +(U<NUMX)/LOOP1
[13]   U+0
[14]   V+V+1
[15]   +(V<NUMY)/LOOP1
    V

    V VIEW1 OBS
[1]  A COMPUTES CRITERION FOR RANGE DATA
[2]  A USES CRITERION DEFINED BY 'TEST' V
[3]    NUMX+L(LX-LTEMPX)+STEP
[4]    NUMY+L(LY-LTEMPY)+STEP
[5]    CRITERION+(NUMX.NUMY),0
[6]    U+V+0
[7]  LOOP1: SAMPLE+OBS[(U*STEP)+,LTEMPX;(V*STEP)+,LTEMPY]
[8]    CRITERION[U;V]+(.TEMP) CORR,SAMPLE
[9]    U+U+1
[10]   +(U<NUMX)/LOOP1
[11]   U+0
[12]   V+V+1
[13]   +(V<NUMY)/LOOP1
    V

    V Z+WHITE STGMA;S
[1]  A COMPUTES GAUSSIAN WHITE NOISE FIELD OF S.D. STGMA
[2]  A AND OF SIZE LX*LY
[3]    S+0
[4]    Z+(LX.LY),0
[5]  LOOP: Z+Z+?(LX.LY),100000000
[6]    S+S+1
[7]    +(S<12)/LOOP
[8]    Z+SIGMA*6+1E-8*Z
    V
```

9. Instructions for Using Software Package for Scene and ATR Simulation

0. Introduction.

This software package, written in APL, is intended for the following purposes:

- a) to simulate terrain, placing of objects, and the noise of sensor
- b) to compute optimal selection of shape elements to characterize a given ensemble of objects: estimate a Lebesgue set in the space of shape elements
- c) to test for presence of objects in terms of their shape elements
- d) to integrate the results of the tests by a parallel logic algorithm for the detection and recognition of objects against clutter.

1. SIMULATION OF DATA

Choose size of picture (total size) as LX by LY. For example LX=LY=68 or 128. This is only limited by the size of the workspace that is available.

Choose vertical dimension of sky, LSKY, as upper part of picture. For example, LSKY = 12.

Choose height scale ZSCALE. This is only for convenience: to be able to make landscape feature large or small compared to object size after objects have been defined. For example, start with ZSCALE = 1.

1.1. GENERATE OBJECTS. First define, for each object and orientation the viewed object boundary by a matrix OBJECT0, OBJECT1, OBJECT2, etc. with two columns; x and y values in respective columns. Here x and y axes have conventional orientations (not as in matrix coordinates where first coordinate

points downward and second to the right). Call, for the first object,

OBJECT0 ~ GENOBJECT NPOINTS

where NPOINTS is number of corners in polygonal boundary. The user will be interrogated for the x,y values of the corners. Result is the desired corner matrix.

First, set size L×L of object frame. Here L will be smaller than LX and LY. Example: L=37 or 51. A prime number is convenient since that will avoid a special condition when executing function INSIDE called by GENOBJECT.

After all object types have been defined, say NOBJECTS of them, collect all of them into a 3-dimensional array ALLOBJECTS by calling, after defining LSCALE, relating coordinates in polygonal boundaries to pixel size in L×L picture (for example =.75) the function

COLLECT

Note: The number of objects in a simulated scene can be smaller (some object types do not occur in the scene) or larger (some object types occur more than once in the scene) than NOBJECTS.

To display a picture (quickly and crudely on the screen) given by a binary matrix defined digital picture IMAGE with the conventional orientation of coordinate axes called the function

SEE IMAGE

Which was the character □ for an inside point (=1) and a blank for an outside point (=0). This will give a rough idea of how easy/difficult it will be to discriminate between the objects. Use for example IMAGE ~ OBJECT0 etc.

Note. When we call COLLECT, and later on GENSCENE, the function INSIDE is invoked. Be sure to select x-coordinates that are not multiples of 1/L; otherwise a warning statement is printed to indicate special condition not accepted by the function inside.

1.2. GENERATE BACKGROUND. The simulated landscape will have smooth large scale features but also more chaotic small scale features.

The smaller scale is controlled by a parameter SMALLSCALE, for example =.1.

We now call the function GENSCENE in the format

RANGE ← GENSCENE NHILLS

where the parameter NHILLS controls the number of hills in the large scale features.

This function also needs the parameter BACKGROUNDERROR = standard deviation of the measurement errors of the range sensor, for example =10.

The function interrogates the user about placements of objects and their location as a 2-vector with (x,y) coordinates. Respond to question about which object by inputting digital representation of object, for example type 5:

ALLOBJECTS[5;;]

It calls another function GENTERRAIN that interrogates the user about location of the objects. This function needs the following global variables, with suggested values,

AMIN=4 AMAX=50

BMIN=4 BMAX=50

CMIN=3 CMAX=10 or 5.

During execution the simulated values of A,B,C as well as XO,XY are printed for each hill. The C-value determines the height of a hill, A and B the half axes of their elliptical contours, and XO,YO their location, all expressed in pixels as units.

The resulting LX,LY matrix RANGE contains the measured distances. They are obtained using the function SIGHT1, which needs the depression angle PSI (in radians), for example .2.

1.3. UTILITIES.

1.3.1. To get a crude idea of a range field RANGE execute

DISPLAY RANGE

The resulting character matrix uses the symbols 'blank . * o □' for increasing values of the range.

1.3.2. In the IBM-CMS environment, to make the APL matrix MATRIX into CMS file called MATRIX IMG

do

```
)CLEAR  
 )COPY IOFNS  
 )COPY WS MATRIX  
'MATRIX DATA' PUT (20 FORMAT MATRIX)  
 )OFF
```

1.3.3. To rearrange matrix IMAGE for viewing it with conventional orientation of axis do

NEWIMAGE ~ TURN IMAGE

and to scale it to NLEVELS grey levels

NEWIMAGE ~ NLEVELS NEWIMAGE

for example with NLEVELS ~ 64.

1.3.4. To form the 3-column matrix HLIST needed later, and whose first column is the shape element list of hypothesis no. T, and whose two remaining columns are (x,y) locations of the shape elements, execute

FORMLIST T

At the same time it computes the corresponding TLIST from the image. It results in the global matrices HLIST and TLIST.

1.3.5. To compute the global matrix ELDIST of Hamming distance between shape elements do

DISTEL ~ ELDIST

where ELDIST will be a NGxNG matrix of mutual distances; this is done after FINDGENS, LEBESGUE have been executed.

Use the function SEEGENS to see generators (shape elements) obtained from FINDGENS and LEBESGUE.

2. SELECTING GENERATORS

2.1. SHAPE ELEMENTS. Generators will have to be chosen as shape elements: neighborhoods of objects view through windows of size LTEMPX,LTEMPY centered at points of the boundaries of objects. For example LTEMP=LTEMPY=9.

All the shape elements will first be computed and stored in a 3-dimensional array ELEMENTS. This is done by executing the function FINDGENS which needs the global variable DIST which separates points along boundary of objects,

for example DIST=3. Execute the function as described below

FINDGENS

Note. It is necessary that object boundaries obtained from OBJECTS be at least distance LTEMPX,LTEMPY away from boundary of LxL matrices in OBJECTS.

It is a good idea to display at least some of the shape elements, using the utility function SEE or SEEGENS.

So that, after executing

COLLECT,

which creates a 3-dimensional array called ALLOBJECTS, do

SLICES ~ FINDGENS,

which selects shape elements centered at boundaries of all the objects.

2.2. THE LEBESGUE SET ESTIMATION. To estimate Lebesgue set in space of all LxL shape elements we execute

SLICES ~ K LEBESGUE SLICES

where K is a number of generators, NG, wanted and a global variable RADIUS designs the radius of the balls used to cover the Lebesgue set. RADIUS should be fairly small, say 5-15, the local variable K around 20. (there is also a global K!).

Execute GENHYPLISTS to form hypothesis lists HYPELO, HYPLOCO, HYPEL1, etc.

3. RECOGNITION OF OBJECTS

3.1. TEST LIST. To get the two lists resulting from the tests on the matrix RANGE we execute

TESTS RANGE

This computes a list (vector) of shape elements detected in RANGE and a list (matrix with two columns) of the corresponding coordinates for the centers of these shape elements.

The driver TESTS calls the following functions:

a) VIEW IMAGE

which carries out a test (done by the function named TEST) for a particular shape element. Then looping over all the generators in the $3 \times \text{LTEMPX} \times \text{LTEMPY}$ array. It results in each time in a matrix CRITERION

b) SEARCH CRITERION

which computes local maxima in CRITERION separated by a ℓ_1 -distance DIST

c) IBOUNDARY (to find boundary of a digital image)

d) SCREEN (to select boundary points separated by distance DIST)

e) COORDS (to transform to pixel coordinates)

f) TRANSFUVS (to transform n,v coordinates to x,y's)

all of which are auxiliaries.

g) CUTOFF, use K=6 or smaller.

For TEST the user can select one of

{ STUDENT (computing standardized student t-test)
FISHER (for Fisher's z-test)
KOLMOGOROV (for Kolmogorov criterion)

or he can substitute a test criterion of his own.

A global variable STEP controls how crude the search should be, perhaps 4 or 8.

3.2. Now we can execute the driver

RECOGNIZER

It calls the following functions

- a) EVIDENCE computes degree of evidence for a hypothesis specified by HLIST on basis of data in TLIST. It results in a matrix AGREEMENTS.
- b) MATCH which matches a HLIST against a TLIST. The global vector KEEP is the set of shape elements tested that should be kept after discarding the significant ones relative to the current hypothesis.

The function RECOGNIZER types out objects recognized in the observed image.

3.3. An improved version of the above is the driver in the format

RECOGNIZE IMAGE

which calls the main functions TESTSI and FULLTEST.

If no object is detected in IMAGE nothing is printed. In the opposite case the credibility of each object type is evaluated and printed starting with the most credible hypotheses.

The scene can contain several objects, but, at present, not more than one of each type. This can be generalized by using the vector KEEP and looping through TLIST after the first run modified by dropping all its rows except the ones labelled in KEEP.

4. GLOBAL VARIABLES AND THEIR MEANINGS

<u>name</u>	<u>meaning</u>	<u>typical values</u>
AGREELEVEL	threshold value used in accepting tested shape element	-10 - -5
ALLOBJECTS	3-dim array storing all objects as LxL digital binary matrices	
AMAX	largest half axis of elliptical hill contour	20-50
AMIN	see AMAX, smallest value	0-5
BACKGROUNDERROR	S.D. of sensor error	5-10
BMAX	as in AMAX	20-50
BMIN	as in AMAX, smallest value	0-5
CENTERS	centers of hills generated by function GENTERRAIN	
CMAX	highest hill possible	10-20
CMIN	as in CMAX, smallest value	1-5
CONST		1
COORDS	2-column matrix for coordinates of maxima detected in function SEARCH	
CRITERION	matrix of test values	
DIST	distance along boundary of object, used by functions SELECT and FINDGENS	2-4
DISTEL	NGxNG matrix of Hamming distances between shape elements	
DMAX		32
(ELEMENTS)		
ERROR	error matrix	
FIELD	LxL matrix representing heightfield in function SIGHT1	

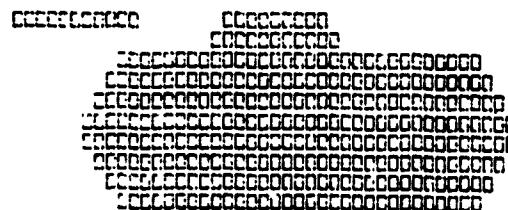
<u>name</u>	<u>meaning</u>	<u>typical values</u>
GENERATORS	3-dim array of shape demands, each of size LTEMPX×LTEMPY	
HLIST	vector of hypothesized shape elements discovered in range data	
HYPELO etc.	shape element list for respective hypotheses	
HYPLOCO etc.	same but 2-column matrix with location of shape elements	
IMAGE INTERIOR	binary digital picture inside of given polygon	1
K	cutoff cont	6 or 9 in respective TEST programs
KEEP	vector of shape element numbers to be kept after testing	
KEEPMAX	final vector KEEP for given HLIST and TLIST	
L	size of object binary numbers (preferably prime)	37 or 51
LSCALE	scale factor used for transforming polygonal shape to fit L×L matrix	.075
LSKY	height of sky in scene	0-20
LTEMPX LTEMPY	x-length of template y-length of template	9-11
LX LY	x-side of picture y-side of picture	128 128
MINDIST	smallest distance assumed between hypothetical locations of object centers	10
NAGREE NAGREEMAX	number of agreements largest number of agreements found	6-8
NB	number of neighbors in lattice topology	8

<u>name</u>	<u>meaning</u>	<u>typical value</u>
NEIGH	minimum separation between recognized local 'maxima'	1-3
NG	number of generators	10-20
NH	number of hypothesis in HLIST	
NOBJECTS	number of object types allowed	4-10
NROWS	number of rows in matrix	
NSLICES	temporary number of shape elements	
NT	number of test values in TLIST	
OBJ	one of the objects	
OBJECTERROR	s.d. of signal from objects	.5
OBJECT0 etc.	objects in digital form	
OFFSET	vector needed to bring shape element to desired position	
POWER	exponent in power law for hill height distribution	1-3
PSI	depression angle	0.1-0.2
RADIUS	radius of balls in shape element space	4-20
RANGE	LX×LY matrix with range data	
SAMPLE	subsort of range data	
SIGMA		1
SLICES	temporary 3-array of shape elements	
SMALLSCALE	s.d. of small scale features of scene-heights	0.1
STEP	size of step in searching range data	4-8
TEST	character vector of test statistics used	'FISHER' 'KOLMOGOROV' 'STUDENT'
TESTSEL	shape element vector selected after testing data	

<u>name</u>	<u>meaning</u>	<u>typical value</u>
TESTSLOC	similar to TESTSEL but 2-column matrix with locations of shape elements selected	
TLIST	vector of shape element numbers selected after testing	
WEIGHT	weight for distance between shape elements relative to distance between their locations	.2-1
ZSCALE	height scale factor	1

10. Objects

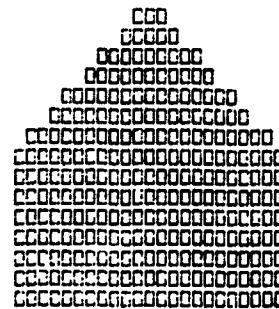
OBJECT NO. 0



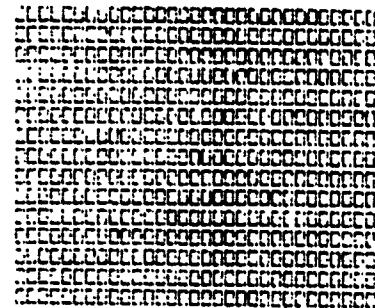
OBJECT NO. 1



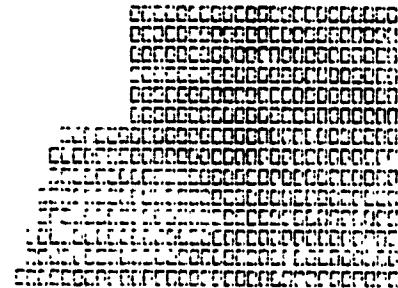
OBJECT NO. 2



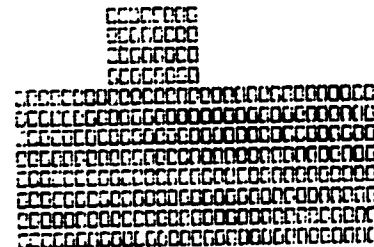
OBJECT NO. 3



OBJECT NO. 4



OBJECT NO. 5



11. Shape elements.

12. Conclusions and continuation. Although the current version of our decision function performs fairly well, at least in unobscured situations, it has some weaknesses that should be removed.

12.1. It has a number of parameters that have to be set before execution. At least two of them are critical: the acceptance percentage K , and the threshold **AGREELEVEL**. One should also investigate by systematic experimentation with the simulator, if the decisions depend critically upon other parameters.

One should make the algorithm adaptive in the sense that parameters are either present, when this works well, or estimated from the picture. The function **CUTOFF**, which was used in the earlier version **TEST1** and **RECOGNIZER**, is an example of what sort of estimation procedure to use.

12.2. The final matching done by **RECOGNIZE** uses **FULLTEST** that employs a naive algorithm, trying all combinations between **TLIST** and the **HLIST**'s. This can amount to up to 4000 comparisons. Since, at that stage of the computing, we have a good idea of where hypothetical objects may be located, it should be enough for each object to let **OFFSET** take on about 10×10 locations. For a scene with two objects, and with some false positives presented by **TESTS**, this would amount to about 300 values for **OFFSET**. This modification should therefore lead to speed up by a factor of 10 or so.

This is imperative in realistic situations, where the number of object shapes could easily be 500 or so. The matching will then be done by adding a hashing component to **RECOGNIZE**.

12.3. The experiments have, so far, included only a few cases with obscuration. This should be continued systematically in order to find more precisely under what circumstances the decision function does not work well. Such knowledge is likely to lead to further improvements.

12.4. When real pictures become available in larger numbers the decision function should be tested on them. If the technology has improved to such an extent that signals returned from the background carry more information, this should be reflected by changes in line TESTS[20]. It may, at such a stage, also be meaningful to replace our flat shape elements by spatial shape elements.

Appendix 1

Micro-statistical Analysis of Noisy Images

1. The micro-statistical approach. A major advantage of stochastic relaxation is its universality: it can be applied, at least in principle, to most pattern inference problems based on Bayesian models. Its use requires, however, a massive computing effort, especially when the underlying connector graph σ is large and the couplings expressed by the prior measure are strong.

We have of course been aware of this in our group from the very beginning. Since our algorithms are of parallel type one can hope that they are still feasible in realistic situations given that parallel hardware is available. But we have also tried to achieve computational feasibility by analytic studies that have led to pattern theoretic limit theorems, most recently in work by Chow, Grenander, and Sethuraman.

In this report we shall suggest another analytic attempt to reduce the computing drastically. It is based on the micro-statistics of the Markov process induced by the prior: local statistics defined in small sets surrounding the sites in the connector graph. The size of these sets should be much smaller than n , the number of sites, but considerably larger than w , the size of the neighborhoods in the graph topology.

The micro-statistical approach will lead to fast computing when compared to stochastic relaxation, but we pay for this by sacrificing some global information as will be mentioned in Section 3. The optimality criteria, on which the algorithms will be based, will be local so that we shall deal with another optimality criterion than the MAP methods.

But computational speed is not the only issue here. We know (see the preceding report in this series) that sometimes we are forced to use local methods, for example when we allow occlusion in the scene, or when the objects have internal degrees of freedom. It is then natural to apply the micro-statistical approach.

To illustrate the approach some computer experiments have been carried out, so far only in very limited situations, by Johan Berglind. It is too early to make any definitive claims, but the experiments at least established the feasibility of micro-statistical algorithms in the special circumstances examined.

2. Formalizing micro-statistical concepts. To fix ideas, say that the connector graph σ is a square lattice \mathbb{Z}_L of size $L \times L$ in the plane and denote its sites by $z = (x, y)$, where $x, y = 0, 1, 2, \dots, L-1$. Further we assume the lattice to be periodic and have the usual closest neighbor topology, $\omega = 4$. The pure images I will be binary, $I(x, y) = 0$ or 1 , while the deformed (observed) images I^D may be B/W or gray level, $I^D(x, y) \in C$, with a background space C of arbitrary cardinality. For simplicity we also assume that the noise, conditioned by I , is i.i.d., so that it is enough to specify a conditional density p_n

$$(1) \quad \begin{cases} p_n(a, b) = p[I^D(x, y) = a | I] \\ b = I(x, y) \end{cases}$$

None of these assumptions is essential for the following and can easily be extended to more general situations.

We shall think of the pure images as discretized versions of sets in \mathbb{R}^2 with boundaries that are piecewise smooth, and where the length scale $1/L$ is small. The points $\xi = (\xi, \eta)$ of the unit square (actually 2-torus) with $0 \leq \xi, \eta < 1$ are made

to correspond to the lattice points 2 of the digital image

$$(2) \quad \begin{cases} \xi = x/L \\ \eta = y/L \end{cases}$$

2.1. Look at the scene in \mathbb{R}^2 through a square window $W_{\xi\eta}$ centered at (ξ, η)

$$(3) \quad W_{\xi\eta} = \{(\xi', \eta') \mid |\xi' - \xi| < \delta, |\eta' - \eta| < \delta\}.$$

We then see a slice of the pure digital image

$$(4) \quad \begin{cases} I_{xy} = \{I(x', y') \text{ with } |x' - x| < \delta L, |y' - y| < \delta L\} \\ x = \xi L, y = \eta L \end{cases}$$

if ξ, η are multiples of the length scale $1/L$.

How will these slices look for small δ and large L ? Three cases can occur:

1. I_{xy} is identically 0, all its points outside of the objects.
2. I_{xy} is identically 1, all its points inside the objects.
3. I_{xy} consists of points both inside and outside the objects.

In the third case a boundary separates points inside and outside. This boundary is, asymptotically straight, unless we happen to be at a corner point of the object - then the boundary is asymptotically that of the outside or inside of a triangle (unless the smooth boundary arcs meet at angle = 0° or 180° which will be ruled out for convenience). Higher order approximations may be required.

2.2. Introduce now formally a set U of slices, a slice, generically denoted u , being a subset of a square $Q = (-1, 1) \times (-1, 1)$. The boundary separating 0 from 1 in Q of a slice is called its divider. For any $n \in U$ we consider the set I_{xy}^u of pure images $I \in \mathcal{F}$ for which the restriction to the lattice points (x', y')

$$(5) \quad |x' - x| < \delta L, |y' - y| < \delta L$$

we have

$$(6) \quad I(x', y') = u \left[\frac{x' - x}{6L}, \frac{y' - y}{6L} \right]$$

Here we use the symbol u also to denote the indicator function of the set u .

A prior P on \mathcal{F} will induce a probability measure \bar{P} on the set $\mathcal{F}_{xy} = \{I_{xy}^u | u \in U\}$ in the following way. If U is finite we introduce

$$(7) \quad \bar{P}(I_{xy}^u) = \frac{P(I_{xy}^u)}{\sum_{u \in U} P(I_{xy}^u)},$$

assuming of course that the denominator in (7) is not zero. If U is a subset of some \mathbb{R}^d we introduce \bar{P} by densities, say with respect to Lebesgue measure in \mathbb{R}^d , in analogy with the expression in (7).

\bar{P} is our micro-statistics and we shall use the above construction, with appropriate modifications as needed, in several cases.

First let us consider some slice families U .

Ex.1. Let U consist of all sets obtained as the intersection of Q with a half-plane. In particular $u=0$ shall mean the empty set and $u=1$ the full set Q . The divider will thus be empty or a straight line segment.

Ex.2. Let U consist of all sets obtained as the intersections of Q with the union or intersection of two half planes. Then U contains all the slices from Ex. 1 but also wedges. The wedges will approximate the local behavior of a boundary point of an object where the tangent direction jumps.

Ex.3. The two previous slice sets have boundaries consisting of zero, one, or two line segments. An obvious extension is obtained by letting the divider be a conic section, or two conic arcs joined together.

2.3. We are now ready to describe the local restoration procedure based on the micro-statistics family U . Consider an arbitrary site $z = (x, y)$ and form the joint frequency function f of the micro-statistic u and the observed image in the window centered at z (solve (6) for $u(\xi, \eta)$)

$$(8) \quad f = p(u) \prod_{\xi, \eta} p_n[I^D(x+6L\xi, y+6L\eta | u(\xi, \eta))]$$

where the product is over the window used.

The expression in (8) assumes a discrete U -family of slices. If U is continuous a factor for the Jacobian should also be needed.

Now make the decomposition

$$(9) \quad U = U^0 \cup U^1$$

where U^0 consists of all slices with $u(0,0) = 0$ and U^1 contains the ones with $u(0,0) = 1$. Introduce the probabilities

$$(10) \quad \begin{cases} \pi_0 = \sum_{u \in U^0} f \\ \pi_1 = \sum_{u \in U^1} f \end{cases}$$

and use as the optimality criterion for the restoration minimum expected Hamming. The resulting algorithm for restoration based on the micro-statistical family U is then simply

$$(11) \quad I^*(x, y) = \begin{cases} 0 & \text{if } \pi_0 > \pi_1 \\ 1 & \text{if } \pi_0 < \pi_1 \end{cases}$$

and randomized $I^*(x, y)$ in the exceptional case $\pi_0 = \pi_1$.

2.4. Let us look at this algorithm for the special cases in 2.2. If we denote the length of our digital square $28L$ by m we get the following

Ex.1. The cardinality $|U| \sim 12m^2$ (enumerate the two points of intersection between the divider and the boundary of the digital square), so that if $m=5$ we get about 300 slices. We precompute the $p(u)$ values for these slices. For each of these slices we must compute, for all $z=(x,y)$, the product in (8), with $m^2=25$ multiplications. Thus, for each site we need about 7500 operations. The step in (11) consists of just a single comparison. Note that with this approach *all the sites can be processed in parallel*. If, for example, $L=256$, we need order of 1 sec on the STAR. If, instead, truly parallel hardware is available execution time is probably negligible. And this is with no attempt at speed up of our algorithm!

Ex.2. Now when the slices also include wedges, so that our algorithm is a boundary-and-corner detector, we get a larger slice family U , 25 times as big (the corner point of a wedge has m^2 possible positions) if all wedges are included. The need for parallel hardware is then accentuated unless algorithmic speed up can be achieved (which seems likely). In the pure image we would normally expect most points to be internal or external, fewer would be on a boundary, and very few would be corners. Perhaps this can be exploited for speed up at the price, naturally, of more complicated program logic.

Ex.3. When we allow more general form for the divider the cardinality of the slice family U will increase. This changes little in principle but at present it is unclear whether the increase in restorative power motivates the greater computing effort that will be required.

Actually, the issue here is related to the discussion in the preceding report in this series dealing with the 'car experiment'. It is only when we have detailed information a priori concerning pattern structure that it will pay off to base the

analysis on micro-statistics parametrized by boundary arcs represented by dividers forming a larger U-space than in Ex. 1 and 2.

2.5. Let us make some additional remarks on the implementation of the idea of micro-statistics.

Remark 1. The choice of prior on U can of course be derived from the prior on \mathcal{T} . This was done, at least approximately, in Berglind's computer experiment using the Ising model. It could also be done for more interesting image models.

It is tempting, however, to choose the measure on U directly, and perhaps estimate it directly from data. This has not been tried yet but should be examined analytically and experimentally. If we do this we should also pay attention to the problem whether such a specification is consistent, in the sense that the slice distribution p on U can be obtained as a local approximation of any full prior P on \mathcal{T} .

Remark 2. Consider for a moment the deformation mechanism D where inside points result in $N(m_0, \sigma^2)$ values and outside points in $N(m_1, \sigma^2)$ values. In practice m_0 and m_1 are unlikely to be known a priori and we have studied how they can be estimated, for example using the half variance method.

However, we could allow the m 's to vary slowly over the image, and estimate them locally for each slice u . This looks like a realistic alternative, at least when the slice window is large enough so that the statistical variability of the estimates is small enough.

Remark 3. We have limited our discussion so far to set patterns: the pure images are binary. Nothing prevents us, however, from applying the micro-statistical approach to contrast patterns: the pure images take contrast

values in a space of higher cardinality. The slices could then be, for example, polynomial approximations to $I(\cdot, \cdot)$ obtained from a Taylor expansion.

Remark 4. Finally an idea that is only speculative. For each location z we use micro-statistics from its associated window, this is done separately for all locations. This is attractive, both because it follows from an optimality reasoning and because it can be implemented in parallel. Nevertheless, it may be possible to gain in restorative power by linking overlapping slices together, let them be coupled. Is there anything in this idea? I feel uneasy about it since it contradicts our general strategy to derive model based methods - linking slices together is more of an ad hoc idea.

3. **Summary.** The advantages of the micro-statistical approach are a) it is single phase (not iterative with questionable speed of convergence), b) it seems to reduce computing drastically (note that it avoids simulation), and c) it is based on an optimality criterion that is, at least to some extent, meaningful and natural.

The main disadvantage that I see in it at present occurs when c) is not true. Indeed, we give up some global cohesiveness, and for this reason the restored images may simply not "look right". Only further study and systematic experimentation will show if and when this is a serious objection. This is of course related to Remark 4.

Appendix 2

Parallel Logic Under Uncertainty. Continued and Applied to the "Car Experiment"

1. Present status of problem. The study carried out by our group is object identification, to be implemented in the car experiment, has dealt with a number of issues and it is time to take a look at what we have done. The following list is not complete but will help to clarify the situation and plan the continuation.

1.1. We have assumed that the scene contains one or no object of a finite number n_α of prescribed types, denoted $\alpha; \alpha=1,2,\dots,n_\alpha$. The scene is viewed at a distance r and from an elevation angle ψ , both of which are assumed known.

1.2. The object is assumed to be rigid and opaque so that it has no internal degrees of freedom. Call the location z in the object plane, where z is some distinguished point, for example the centroid of the object. Also denote the orientation of the object by ϕ , where ϕ is the angle between some distinguished direction of the object and a fixed orientation in the object plane. Hence $z \in \mathbb{R}^2$, $\phi \in [0,2\pi]$.

1.3. Observing the scene we denote the view by I^D , where I^D is a digital $L \times L$ pixel, grey level picture in the image plane orthogonal to the line of sight. The distance r is assumed big enough so that we can limit ourselves to orthogonal projections, but this is easy to modify if it turns out to be desirable. The deformed image I^D represents in addition to the object, if any, noise from the observational set up and a noisy background (clutter). At present we model this simply by two i.i.d. samples from Gaussian random variables $N(m_\nu, \sigma^2)$ where $m_\nu = m_0$ inside the object area and $m_\nu = m$, outside. This will probably be

replaced by two stationary, but not necessarily isotropic, Gaussian stochastic processes for object and background. To handle this extension analytically we have discussed, at an early stage of our work, to use Toeplitz approximations; this can wait since we are familiar with such mathematical techniques.

1.4. We have developed boundary detectors using Bayesian estimators derived from various Markovian models of the shape of the object. The simplest of these, the Ising model, expresses only clumping tendencies. While it is of theoretical interest it includes too weak prior information about object shape to be a serious candidate for our continued work.

Two other lattice based models; closely related to each other, express more detailed prior shape structure; one is the pixel-edge model, the other is built on generators representing geometric tendencies of the boundary.

Still another model is continuum based and expresses the boundary as a spline, in the simplest case reducing to a polygonal boundary.

The parameters m_0, m_1, σ^2 cannot be assumed to be known in advance. To deal with this we proposed and studied the half variance method, which seems to work fairly well but needs some work.

1.5. The experimental set up uses small model cars, painted white and, so far, located on a black background. The digital image is processed using various boundary detectors, in order to get a better feeling for how the estimates of the boundary behave statistically.

1.6. The work undertaken by the Hughes group should be of great help in this. A careful data analysis of the results should make it possible to firm up the mathematical models to be used as we go along in our joint work with them.

In an early stage of our study we discussed the possibility of applying stochastic relaxation to a vector of selected statistics computed from an estimate

of the boundary. At that time we were thinking of statistics such as the area of the convex hull, the degree of concavity, the lengths of diameters, and many others. The result of a successful data analysis should help pinpoint good such statistics. This can be viewed as the selection of hash functions, but the noisiness of data makes for an unconventional form of hashing. As will be suggested below we may be able to replace this by a less arbitrary approach.

1.7. We have pointed to the important correspondence problem between the true and estimated boundaries. This is a case of unlabelled observations.

To deal algorithmically with the correspondence problem we discussed a dynamic programming approach. Since it seems to require a massive computational effort we may have to look for substitutes involving less computing.

1.8. We have studied the problem of optimal image approximation when a piecewise smooth boundary is approximated by a polygon or possibly by other splines. The analytical result has been tried on real pictures and seems to work well. The result will be applied systematically later on, when we shall use it to obtain data compression when storing object profiles.

2. Research strategy. We have not yet attempted to put these pieces together to obtain an integrated solution to our problem in object identification. I think it is time to start doing this during the academic year just begun. As a preparation let us examine the general approach we have been using, explicitly or not.

2.1. A guiding principle for our work has been the model based paradigm: algorithm for pattern analysis should be derived from clearly stated mathematical models. Not only should the patterns, say shapes of objects, be described mathematically in detail, but the same should hold for the whole chain involving patterns and their generation, the observer, the deformation mechanism, and the

algorithms for inference. The links in this chain should be integrated.

2.2. Another tendency is reductionism: the model construction should be reduced, as far as possible, to first principles. In the case of our car experiment it means that we should emphasize the mathematical generation of shape, the pure image, this should be the starting point.

How to do this depends upon the degree of variability of the shapes encountered. In the car experiment as envisaged so far we have only three degrees of freedom, two for z and one for ϕ . The nuisance parameter is hence low dimensional, $\theta = (z, \phi) \in \mathbb{R}^3$. As will be suggested below we should go along to higher dimensional nuisance parameters which will force local methods upon us, and we shall factor our problem into many small ones of low dimensionality; this will be made clearer below. In a related study, the leaf shape project, we meet the extreme of this, when the θ -space is infinite, or at least very high, dimensional.

2.3. We also aim for generality. The choice of the car experiment is motivated not only by practical considerations, but also for its concreteness. Nevertheless, we hope that the results will extend to much more general situations.

To make this possible I suggest that we extend the problem to allow for several objects and allow occlusion. This will increase the number of d.f. We should also allow internal d.f.'s, for example nuisance parameters representing relative angles of parts that can be moved, say of doors, wheels, etc. Or, parameters expressing the distribution of albedo, temperature, etc. over the object surface.

Perhaps this sounds too ambitious at the present time, but we shall outline a methodology for inference intended for such, more general pattern structures. It is based on the idea of parallel logic under uncertainty that I have suggested elsewhere.

3. Formalizing the factoring of the problem. Now let us be more specific. In order to achieve greater descriptive power we shall use a continuum based approach, not a lattice based. The generators will be chosen as directed analytic arcs, have arity two, with in-bond equal to the start point of the arc and out-bond equal to the endpoint. The analytic form of the arc will be left open until the completion of the data analysis based on the experimental data collected by the Hughes group.

The natural similarity group S is here simply the translation group in the image plane. We shall not use the full Euclidean group, since rotations of the object in \mathbb{R}^3 around a vertical object does not correspond to rotations in the image plane; instead of simple rotations we meet more complicated transformations of the image.

The bond relation ρ will be taken as EQUAL, meaning concatenation of arcs, and the connection type $\Sigma = \text{CYCLES}$, so that each connector graph σ consists of a finite number of closed cycles. This choice of Σ will be seen to have crucial algorithmic consequences later on.

This regularity $\mathcal{R} = \langle \text{EQUAL}, \text{CYCLES} \rangle$ over G via S defines a regular configuration space $\mathcal{L}(\mathcal{R})$. We shall use a subspace $\mathcal{L}'(\mathcal{R}) \subseteq \mathcal{L}(\mathcal{R})$ consisting of the profiles obtained by placing a finite number k of objects of given types in the image plane; the profiles may overlap or be disjoint.

An object is described by a vector $(\alpha, \theta_1, \theta_2, \dots, \theta_\alpha)$ where the θ 's are the nuisance parameters and we use the convention that $\theta_1=x$, $\theta_2=y$, $\theta_3=\phi$ with $z=(x,y)$. The remaining θ 's represent the internal degrees of freedom.

We shall assume the following construction of the objects; the reason behind this assumption will become clear later. Each object is made up of rigid parts, joined by hinges in such a way that the orientation of any of its parts, say the v^{th}

one, is uniquely defined by the principal orientation angle $\theta_3=\phi$ and one hinge angle ϕ_v . This is a sort of separability condition: only first order interactions between θ_3 and any θ_v , $v > 4$, are allowed.

A similar reasoning will be applied when we get to nuisance parameters representing the temperature or albedo distribution over the surface of an object; this will be dealt with later. Here we note only that boundary statistics will then be replaced by what could be called surface patch statistics.

Let us write

$$(1) \quad \mathcal{L}'(\mathcal{R}) = \bigcup_{k=0}^{\infty} \mathcal{L}'_k(\mathcal{R})$$

where $\mathcal{L}'_k(\mathcal{R})$ is the subset of $\mathcal{L}'(\mathcal{R})$ with k objects. Then $\mathcal{L}'_k(\mathcal{R})$ can be parametrized by the vector

$$(2) \quad (\bar{\alpha}, \bar{\theta}) = (\alpha^1, \theta^1, \alpha^2, \theta^2, \dots, \alpha^k, \theta^k).$$

The value of k in practice is likely to be small. Nevertheless the dimension of the $(\bar{\alpha}, \bar{\theta})$ -space may seem prohibitively large, both for string of the profiles and for the decision algorithms to be developed for inference. Can it be done even for the case we have dealt with so far, $k=1$ and $d=3$?

First, it is clear that the two d.f. for $\theta_1=x, \theta_2=y$ cause no trouble. Indeed, we need only store the profiles in standard location, say $z=0$, and the decision algorithm can be based on differences of arc points in \mathbb{R}^2 , relative, not absolute coordinates. For $\theta_3=\phi$ we could have done something similar if it had corresponded to rotations in the image plane. Since this is not so we must store enough information to describe the profiles for different θ_3 -values. Perhaps $\theta_3=0, 10^\circ, 20^\circ, \dots, 350^\circ$ will be enough practically, which means 36 θ_3 -values.

Now let us consider another d.f., θ_4 , say an internal angle. The moveable parts of the objects have been assumed to be joined by hinges whose angles are the θ_v ; $v=4,2,\dots,d$. Obviously, we can not hope to store profiles for all discretely chosen combination of $(\theta_3, \theta_4, \dots, \theta_d)$ but this is not needed. This is where factoring comes in. Indeed, it is enough to store outlines for the rigid parts for a sufficient number of combinations (θ_3, θ_v) , where $v \geq 4$. Perhaps we will have to store 36^2 , but not 36^{d-2} , profiles for each arc: the problem has been factored, reducing storage drastically.

We shall need a prior on the configuration space $\mathcal{L}'(\mathcal{R})$ as always for a Bayesian approach. Since we have widened our problem by allowing several objects, k can be larger than one, and by letting objects have internal degrees of freedom, d can be larger than 3, so that the prior lives on a larger parameter space. The choice of prior speeds more discussion, but at this time let us only consider the probability distribution of k , q_k for $k \geq 0$.

We do not see any reason for any particular analytical form for q_k in general, but only note that it expresses the principle of Occam's razor if its tail is short. Say, quite arbitrarily, that we use a geometric distribution

$$(3) \quad q_k = \frac{1}{1-p} p^k; \quad k=0,1,\dots$$

for the present and leave the final choice till later. The value of p should be small.

The ideal observer cannot always see the whole configuration $c \in \mathcal{L}(\mathcal{R})$, since objects, or parts of objects may hide each other. That means that the identification map $R: \mathcal{L}'(\mathcal{R}) \rightarrow \mathcal{T}$ will only preserve boundary information of the visible profiles.

The deformation mechanism $\mathcal{D}: \mathcal{F} \rightarrow \mathcal{F}^D$ will be chosen as before, two i.i.d. Gaussian samples from inside and outside the pure image I .

4. Choice of inference algorithm. The fact that we now allow k and d to be greater than one and three respectively has an important and obvious consequence for the construction of the inference algorithm. It is clear to anyone who has considered the multi-object situation with partial occlusion, that are now led to base the inference on local statistics. Simple template matching, computationally feasible or not, will not work, since we cannot be sure that we are dealing with one and the same object throughout. Instead we have to break up the estimated boundary, say $(\partial I)^*$, into pieces and extract local information from each piece.

4.1. I am not going to argue against this, but would like to point to an obscure link in the reasoning.

Our deformed image I^D is an $L \times L$ digital, grey level picture. When we base the inference on the estimated boundary we really employ a plausibility argument implicitly. In the spirit of our whole approach we ought to give this an analytical foundation and, at the same time, make the statement mathematically precise.

The above will become clearer if we, only momentarily, go back to the original case $k=1$, $d=3$. Consider a boundary detector ∂^* , an estimator of ∂I , so that $\partial^*: \mathcal{F}^D \rightarrow (\partial I)$. If ∂^* could be shown to be a sufficient statistic we would be on firm ground, but I am convinced that this is not the case (this should be shown rigorously).

However, we may be able to make an asymptotic statement, almost as strong, in terms of the linear length scale $\epsilon = 1/L$. Can one show that some ∂^* is ϵ -sufficient when $L \rightarrow \infty$? Someone in our group ought to settle this question. It may turn out that some boundary detectors ∂^* are ϵ -sufficient, or contain,

asymptotically, all the relevant information in the sample in terms of some other formalization, while others are not, and this would have practical consequences for the design of the inference algorithm and choice of δ^* .

4.2. Let us now return to the general problem, $k \geq 0$, $d \geq 3$. Say that we have decided to use a particular estimator δ^* and we have obtained a boundary $\delta^* I^D$, consisting of one or more closed curves. If there are several topological components in $\delta^* I^D$, let us consider a single one of them at first.

The directed and closed curve, ζ^* , is hoped to contain, asymptotically all the relevant observations in the sample. Divide ζ^* into N arcs of equal length, say $\zeta_1, \zeta_2, \dots, \zeta_{N-1}$ and consider the j^{th} one ζ_j^* . Some hints for choosing N will be given later.

The local information contained in ζ_j^* shall be calculated as follows. Return to a pure (undeformed) image I in \mathbb{R}^2 , consisting of a single object. Consider a directed arc ζ belonging to the boundary of these objects. Parametrize ζ as $h = (\alpha, \theta, s_0, s_1)$ where α is the pattern type of the object, θ its nuisance parameter and s_0 and s_1 the start and endpoint of ζ measured in arclength from some conveniently chosen point in the boundary of the same object α and same θ -value. This h will be our local hypothesis, $h \in H$ where H is the full hypothesis space. Note that it is a simple hypothesis.

4.3. Now go back to the deformed image and the directed arc ζ_j^* . Introduce the probability that an arc ζ gives rise to ζ_j^* as

$$(4) \quad Q(\zeta_j^*, \zeta) = P(\zeta \rightarrow \zeta_j^*).$$

To avoid misunderstanding let us emphasize that ζ stands for an arbitrary arc of the boundary of an object profile, it need not coincide with one of the full analytic arcs generating the spline.

Also let us mention for later reference that it may very well happen that ζ_j^* originates from two object boundaries, one object occluding the other. This is more likely to happen if ζ_j^* is long, or N is small, so that this indicates that one should choose N moderately large, perhaps 15-25.

For a specified estimator $\hat{\theta}^*$ it should be possible to calculate analytically the probability in () asymptotically, but I suggest that we postpone this for the time being.

If and when we can determine Q asymptotically we shall introduce the local hypothesis

$$(5) \quad h(\zeta_j^*) = \{\zeta \mid Q(\zeta, \zeta_j) \geq \delta\} \in H$$

for some threshold value δ . However, I propose that we use, tentatively, the following heuristic way of determining the local hypotheses. Give ζ_j^* with endpoints z^0 (start) and z^1 (end) find the set $\bar{h}(\zeta_j^*)$ of ζ 's that have the same endpoints (counted in the same direction!). Since our similarity group consists of translation of \mathbb{R}^2 it is enough to search for an arc ζ such that the vectors $z^1 - z^0$ and ζ (endpoint) - ζ (start point) are equal. How to realize this algorithmically in a feasible way deserves careful attention, but it is clear that this is much less demanding computationally than the dynamic programming approach discussed in our group. If it is a good way is another matter. For an arc $\zeta \in \bar{h}(\zeta_j^*)$ with length parameters s_1 and s_2 introduce a distance measure like the following, perhaps in a modified form,

$$(6) \quad d(\zeta_j^*, \zeta) = \sum_{v=0}^{m+1} |z_v^* - z_v|$$

where $m+1$ is the number of lattice points z_v^* on ζ_j^* and z_v are m equidistant points on ζ . The exact form of the distance criteria is not important at this preliminary stage, but will be later. What is important, however, is that d should

measure distance between directed arcs, and not be of the form of a Hausdorff (or similar) distance between sets.

Then let us, again tentatively, replace $h(\zeta_j^*)$ by

$$(7) \quad \bar{h}(\zeta_j^*) = \{\zeta \mid \zeta \in \bar{h}(\zeta_j^k), d(\zeta_j^*, \zeta) < \eta\} \subset H$$

with another threshold constant η . The subsets $\bar{h}(\zeta_j^*)$ of H will be thought of as our plausible local hypotheses. Note that they are composite hypotheses in contrast to the pure hypotheses h mentioned earlier.

From now on we shall discard arc length information, the s_0, s_1 values in h . Instead we project the local hypothesis to the space H' consisting of $(\alpha\theta)$ -vectors, say h' , only. Denote the projected set thus obtained from $\bar{h}(\zeta_j^*)$ by $h'(\zeta_j^*)$.

Recalling that our adopted connection type $\Sigma = \text{CYCLES}$, it is clear that our inference problem deals with estimating an unknown connector σ in this set Σ . Think of the arcs $\zeta_j^*, j=0, 1, \dots, N-1$; as the nodes in σ , not as the segments, and use acceptor functions

$$(8) \quad A(h'_{j_1}, h'_{j_2}) = \phi[d'(h'_{j_2}, h'_{j_1})]$$

where ϕ is a positive decreasing function and d' an appropriate distance in H' .

We shall now try h maximize the criterion

$$(9) \quad C(\sigma^*; h_0, h_1, \dots, h_{N-1}) = q(k^*) \prod_{\sigma^*} A(h'_{j_1}, h'_{j_2})$$

where $\sigma^* \in \Sigma$, $k^* = \text{number of cycles in } \sigma^*$, $h_j \in h'_j$, and the product is taken over segments (j_1, j_2) in σ^* . To achieve this we apply stochastic relaxation with annealing to $C(\sigma^*; h_0, h_1, \dots, h_{N-1})$.

Let us mention in passing that the way we have outlined our recognition algorithm the value of k means the number of cycles (wholly or partly visible

objects) in a topological component of I . It does no longer mean the total number of objects in the scene.

Two decisions must be made to implement stochastic relaxation: how to initialize $\sigma^*, h_0, h_1, \dots, h_{N-1}$, and what should be the elementary update operations of the relaxation scheme.

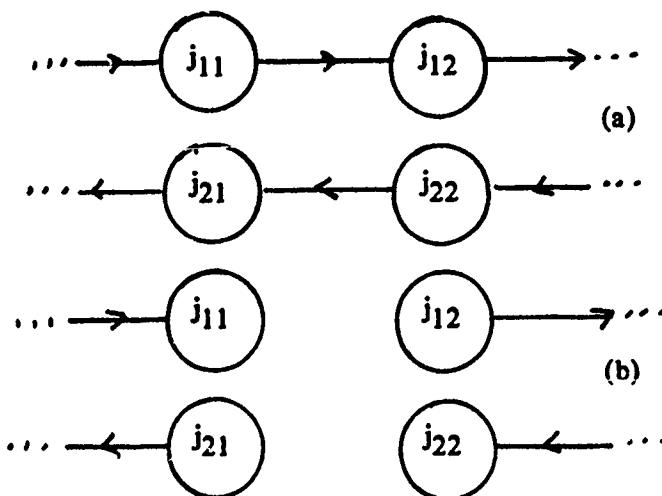
Two suggestions for initialization: 1) Choose initially σ^* as a single cycle, or, 2) Consider the joint plausible hypothesis set

$$(10) \quad \bigcup_{j=0}^{N-1} h_j' \subset H'$$

and use some clustering algorithm to get a rough idea of how many clusters there are in the set. Use a σ^* with this number of cycles distributed as the analysis indicates over the j -values.

The elementary up date operations should certainly involve replacing an h_j' , regarding the rest and σ^* as fixed.

It is less clear what operations we should use changing σ^* . I suggest, as a minimum, that we include cut-and-join operations as indicated in the figure below, which take (a) into (b) or (b) into (a).



The decision on choosing the elementary up date operations must be made in such a way that the class of operations is complete: given two connectors σ_1 and σ_2 in CYCLES there should exist a chain of elementary up date operations that changes σ_1 into σ_2 . I believe that the above choice leads to a complete class.

The conditional probabilities associated with these up dates and (9) are easily calculated, note that k^* may change so that the factor $q(k^*)$ in (9) affects them, implementing a reasoning of type Occam's razor.

Repeating updating, successively lowering temperature, and having iterated many times we stop the process. Our recognition algorithm has as its result that we see k^* objects with α -values as distributed over the current plausible local hypotheses h_j . As a side result we also get estimates for the nuisance parameters.

5. Discussion of proposed algorithm. The presentation in the previous section is only a sketch with many details missing and decisions to be made before the algorithm is completely specified. Nevertheless, I hope that the description is sufficiently clear to serve as a starting point for our continued work.

The approach is basically the same as the one we decided upon a couple of years ago, the main difference being that it does not rest on the solution of a more or less arbitrary set of features characterizing the estimated boundary and the set it encloses. Instead we try to exploit as much as possible of the information contained in the observed image and handle it by a procedure implementing parallel logic for noisy data. It is also more ambitious in that it is intended for multiple object scenes and objects with internal degrees of freedom.

But is it over ambitious? Can algorithms of the type suggested be processed using computer technology available today or in the near future?

For the simplest case, the one to which we have limited ourselves so far, with a single object with no internal d.f.'s, can probably be handled computationally. But does this remain true if we allow more than one object, $k > 1$, and/or internal d.f.'s, $d > 3$? I hope that this is made possible by the factoring of the problem, but at present this is only a guess, perhaps wishful thinking.

The only way to find out if this is possible is to try it on real pictures from our car experiment. I therefore suggest that we do this during the current academic year, at first only for $k=1$, $d=3$, to firm up the model and algorithm, and if the results look promising, that we go on to the general case. I do not mean that we should just carry out a large scale computer experiment, but that we also pay attention to the analytical issues that are only superficially discussed above. Doing this, we may learn that substantial modification is needed, perhaps oversights have to be corrected, and better heuristics invented.

I hope that our group will decide to undertake this ambitious study. We would then have to divide up the work among ourselves to achieve a cohesive team effort.